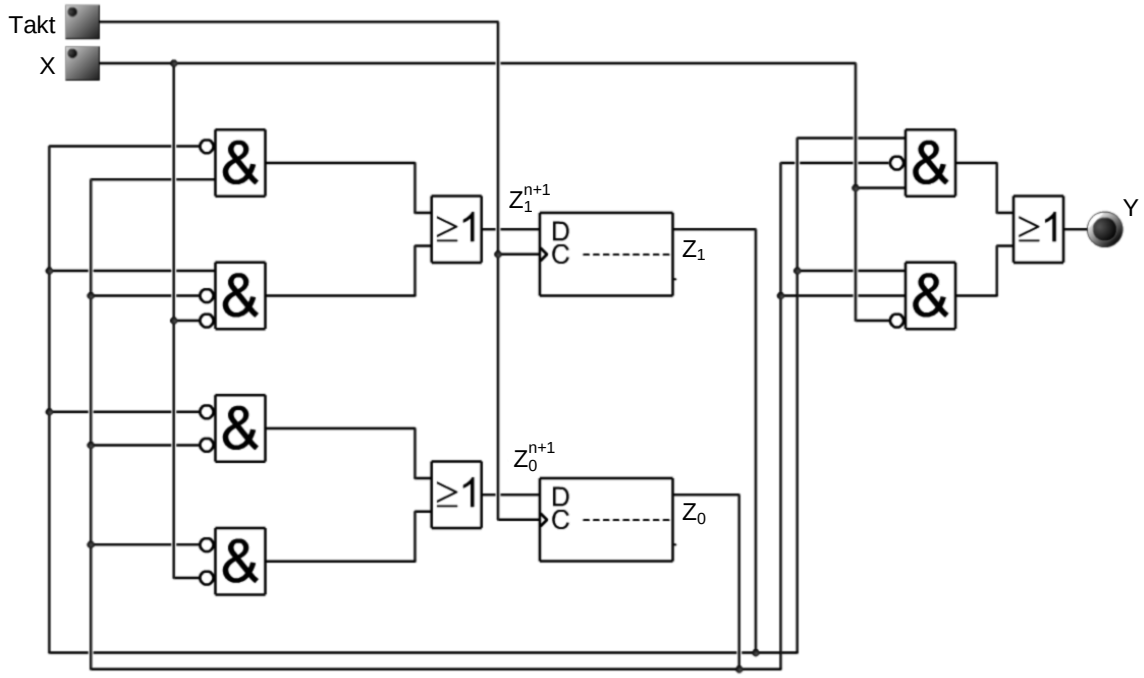
Aufgabe 1.1: Schaltwerksanalyse (12,5 Punkte)

Gegeben Sei folgende Schaltung:



a) Um welchen Automatentyp handelt es sich? *(1 Punkt)*

b) Ermitteln Sie die Ansteuergleichungen der FlipFlops, sowie die Übergangs- und Ausgangsfunktionen des Schaltwerks. *(3,5 Punkte)*

Ansteuergleichungen:

$D_1^n =$

$D_0^n =$

Ausgangsfunktionen:

$Y^n =$

$Z_1^n =$

$Z_0^n =$

Übergangsfunktionen:

$Z_1^{n+1} =$

$Z_0^{n+1} =$

c) Erstellen Sie die Zustandsübergangstabelle! *(3 Punkte)*

	Z^{n+1}		Y^n	
Z^n	$X^n=1$	$X^n=0$	$X^n=1$	$X^n=0$

d) Zeichnen Sie den Zustandsgraphen des Schaltwerks! *(4 Punkte)*

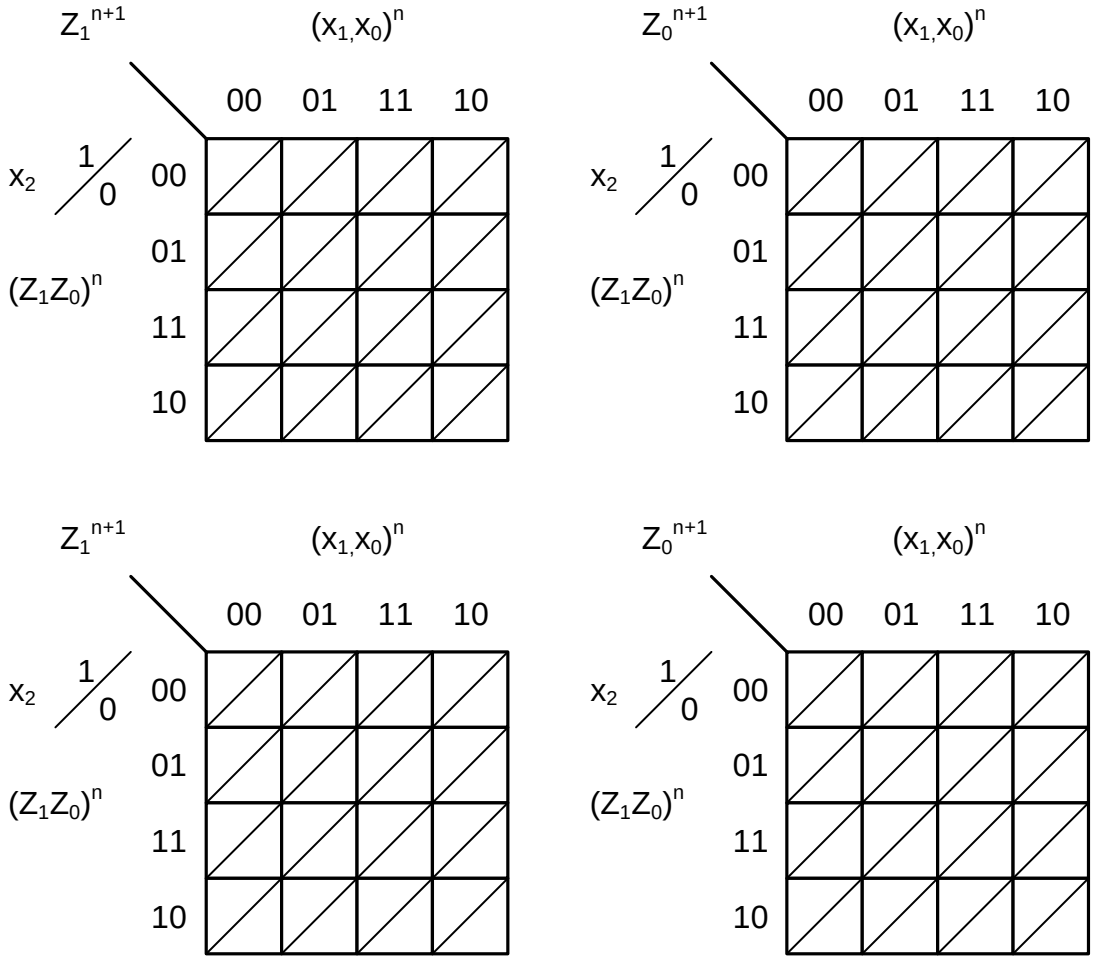
e) Welche Funktion stellt die Schaltung dar? *(1 Punkt)*

Aufgabe 1.2: Schaltwerksentwurf (12,5 Punkte)

Die Zustandsübergangsfunktion eines Automaten mit der Eingabe $(x_2x_1x_0)$, der Ausgabe y_0 und den Zuständen (Z_1Z_0) sei durch folgende Zustandsübergangstabelle definiert:

		$(Z_1^{(n+1)}, Z_0^{(n+1)})$ für die Eingabe (x_2, x_1, x_0)								
Z_1	Z_0	000	001	010	011	100	101	110	111	Y_0
0	0	00	01	00	XX	10	10	01	XX	0
0	1	00	01	00	XX	11	11	01	XX	1
1	0	10	00	11	XX	00	10	11	XX	1
1	1	10	00	10	XX	01	11	11	XX	0

a) Bestimmen Sie die Zustandsübergangsfunktionen Z_1^{n+1} und Z_0^{n+1} als disjunktive Minimalformen mit Hilfe der gegebenen KV-Diagramme. Kennzeichnen Sie die zur Minimierung verwendeten Felder. (8 Punkte)

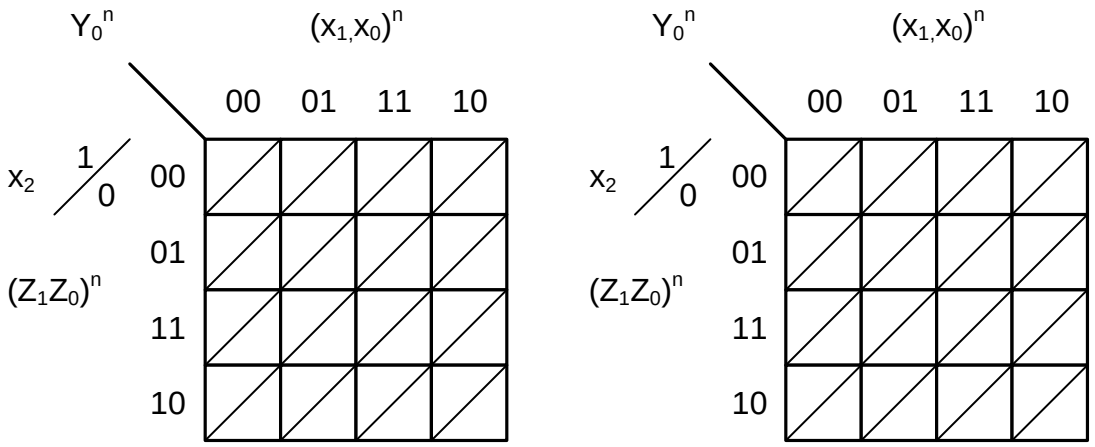


Übergangsfunktionen:

$Z_1^{n+1} =$

$Z_0^{n+1} =$

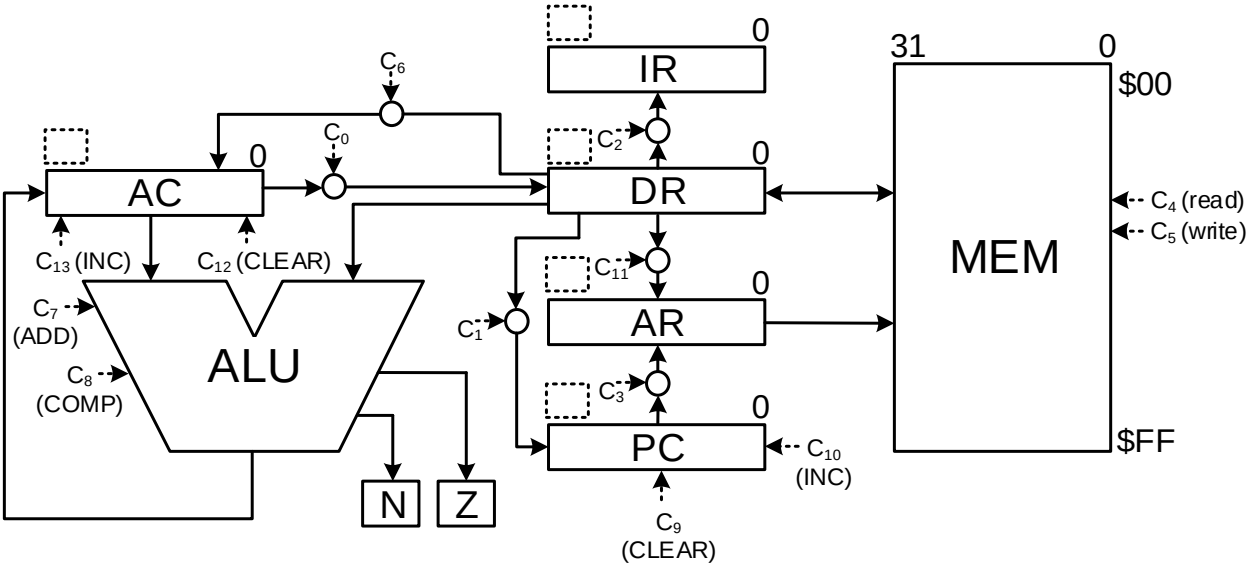
b) Bestimmen Sie die Ausgabefunktion Y_0^n als konjunktive Minimalform mit Hilfe des gegebenen KV-Diagrammes. Kennzeichnen Sie die zur Minimierung verwendeten Felder. (4,5 Punkte)



Ausgabefunktionen:

$Y_0^n =$

Aufgabe 2: Mikroprogrammierte CPU-Kontrolleinheit (25 Punkte)



Blockschaltbild

Das oben vorgegebene Blockschaltbild zeigt das Operationswerk der in dieser Aufgabe zu untersuchenden mikroprogrammierten CPU.

Die ALU beherrscht ausschließlich die Addition (ADD) und das Einerkomplement (COMP). Das Z und N-Flag werden von der ALU automatisch erzeugt und haben 1 Bit Breite. Es gilt: Z=1, wenn die letzte Operation der ALU eine 0 lieferte, sonst Z=0. N=1, wenn die letzte Operation der ALU ein negatives Ergebnis lieferte, sonst N=0.

Eine Registertransferbeschreibung einiger Befehle der CPU ist Ihnen vorgegeben:

Opcode	Befehl	Beschreibung
0	LUI value	Lädt den Wert value an die höherwertige Hälfte Bits des Akkumulators und setzt die untere Hälfte auf null.
1	BNE addr	Setzt den Programmablauf an Adresse addr fort, wenn AC ungleich null ist.
2	BRLE addr	Setzt den Programmablauf an Adresse addr fort, wenn AC ≤ 0 war.

Befehlsformat

a) Ergänzen Sie im Blockschaltbild und im Befehlsformat die fehlenden Indizes der Register in den vorgesehenen Kästen. *(2 Punkte)*

b) Ergänzen Sie in dem auf der nächsten Seite vorgegebenen RT-Programm die frei gelassenen Registerbreiten und –indizes. Ergänzen Sie zudem in den vorgesehenen Feldern (rechts) die Kontrollsignale. *(8 Punkte)*

```
declare register AR(   ), DR(   ), PC(   ), IR(   ), AC(   ), Z, N
declare memory MEM(    )

INIT:       PC <- 0, AC <- 0;                                          _____
FETCH:      AR(   ) <- PC(   ), PC <- PC + 1;                          _____
            read MEM;                                                  _____
            IR(   ) <- DR(   ), switch IR {                            _____
                  case 0: goto LUI                                     _____
                  case 1: goto BRNE                                    _____
                   case 2: goto BRLE };                                _____

LUI:        AC <- 0;                                                   _____
            AC(   ) <- DR(   ), goto FETCH;                            _____

BRNE:       if Z = 0 then PC(   ) <- DR(   ) fi, goto FETCH;           _____

BRLE:       if Z = 1 OR N = 1 then PC <- DR(   ) fi,                   _____
            goto FETCH;                                                _____
```

c) Implementieren Sie die folgenden Befehle selbst und geben Sie auch die Kontrollsignale an. Sie müssen den FETCH nicht weiter anpassen. Das Einerkomplement lässt sich durch **„AC <- not AC"** darstellen. *(3 Punkte)*

STORE addr: Speichert den Inhalt des Akkumulators unter Addresse addr.

SUB addr: Subtrahiert den Inhalt des Akkumulators von dem Wert im Speicher unter Addresse addr und speichert das Ergebnis im Akkumulator.

BRGR addr: Setzt den Programmablauf an Adresse addr fort, wenn das Ergebnis der letzten Operation echt größer als Null war.

d) Erstellen Sie nun ein vertikales mikroprogrammiertes Steuerwerk auf Basis des **vorgegebenen Programms** aus Aufgabe b). **Halten Sie sich, falls möglich, an das vorgegebene Timing.** Es stehen Ihnen ausschließlich die folgenden Condition Select Signale zur Verfügung:

Condition Select	**Funktion**
000	Nicht springen
001	Springe zu der dekodierten Opcode-Adresse (Mapping ROM)
100	Springe, falls Z = 1
101	Springe, falls N = 0
111	Springe immer

Hinweise:

- Es ist in den folgenden Tabellen ausreichend, nur die 1en auszufüllen. Wir nehmen an, dass alle nicht ausgefüllten Felder auf 0 gesetzt sind.
- Sie müssen die Tabellen ggf. nicht voll ausfüllen, einige Zeilen können leer bleiben.
- Verwenden Sie das Ihnen aus der Vorlesung bekannte Mapping ROM.

Gehen Sie wie folgt vor:

(i) Bestimmen Sie eine minimale Kodierung der Kontrollsignale aus b). *(2 Punkte)*

Kontrollsignale														Kodierung			
C_{13}	C_{12}	C_{11}	C_{10}	C_9	C_8	C_7	C_6	C_5	C_4	C_3	C_2	C_1	C_0				

(i) Erstellen Sie nun das vertikale Mikroprogramm in der vorgegebenen Tabelle. Ergänzen Sie den mit den gegebenen Condition Selects umgesetzten RT-Code für BRNE und BRLE. *(7 Punkte)*

addr				CS			Next_addr				c_vert				RT-Code
0	0	0	0												INIT: PC <- 0, AC <- 0;
0	0	0	1												FETCH: AR(7:0) <- PC(7:0), PC <- PC + 1;
0	0	1	0												read MEM;
0	0	1	1												IR(7:0) <- DR(31:24),switch IR {…}
0	1	0	0												LUI: AC <- 0;
0	1	0	1												AC(31:16) <- DR(23:8), goto FETCH;
0	1	1	0												BRNE:
0	1	1	1												
1	0	0	0												
1	0	0	1												
1	0	1	0												
1	0	1	1												
1	1	0	0												
1	1	0	1												
1	1	1	0												
1	1	1	1												

(ii) Geben Sie das Mapping ROM an. *(1 Punkte)*

Befehl	Opcode	Sprungadresse

(iii) Entwerfen Sie analog zum MIPS-Assembler die folgenden Pseudobefehle durch die im Folgenden gegebenen Hardwarebefehle. *(2 Punkte)*

Syntax	**Kommentar**
LUI value	High(AC) = value, Low(AC) = 0
STORE addr	MEM(addr) = AC
ADDI value	AC = AC + value
SUB addr	AC = MEM(addr) - AC
COMP	AC = not AC
BREQ addr	if(AC=0) PC = addr
BRLE addr	if(AC≤0) PC = addr
BRGR addr	if(AC>0) PC = addr

LI 0x1000FFFF: Lädt den 32-Bit Wert 0x1000FFFF in den Akkumulator

BRGE addr: Springt zu addr, wenn AC≥ 0.