# Klausur

# Einführung in die Rechnerarchitektur

## Prof. Dr. Arndt Bode

Wintersemester 2014/2015
4. Februar 2015

Name: _____

Vorname: _____

Matrikelnummer: _____

Geburtsdatum: _____

Hörsaal: _____

Platz: _____

Unterschrift: _____

---

Ergebnis:

| Aufgabe | 1 | 2 | 3 | 4 | Ges. | Note |
|---|---|---|---|---|---|---|
| Punkte | | | | | | |
| Korrektur | | | | | | |

## Hinweise zu den Aufgaben

- Die Bearbeitungszeit beträgt 120 Minuten.
- Es sind keinerlei Hilfsmittel zugelassen, auch keine Taschenrechner.
- Versehen Sie dieses Angabenblatt auf der Titelseite mit Ihrem Namen, Vornamen und Matrikelnummer, sowie Geburtsdatum, Hörsaal und Platznummer. Vergessen Sie nicht die Unterschrift!
- Diese Angabe umfasst *30* bedruckte Seiten (inklusive Deckblatt).
- Außerdem erhalten Sie die folgenden Merkblätter:

  Anlage I: „Die wichtigsten 80386 Befehle und ihre Operanden“

  Anlage II: „Mikroprogrammierung“

  Anlage III: „Die wichtigsten VHDL Konstrukte“

- Alle Lösungen sind in dieses Heft einzutragen. Sollte der vorgesehene Platz nicht ausreichen, so finden Sie am Ende weitere Blätter. Sollten auch diese nicht ausreichen, wenden Sie sich bitte an die Aufsichten.
- Notizpapier wird auf Ihre Anfrage ausgegeben. Die Verwendung von eigenem Papier ist **nicht** gestattet.

*Lösungen auf Notizpapier werden **nicht** gewertet!*

# Aufgabe 1 - Maschinennahe Programmierung

## 1.1 Einzelbefehle

### 1.1.1 Vereinfachung

Fassen Sie die folgenden Befehle zu einem Befehl mit dem gleichen Resultat zusammen.

```
mov al,0
mov ah,1
```

### 1.1.2 Befehlssequenzen

Implementieren Sie kurze Befehlssequenzen, um die genannten Effekte zu bewirken. Temporär genutzte Register müssen nicht gesichert werden.

- AL ins höherwertige Byte von BX und AH ins niederwertige Byte von BX kopieren

- Verschieben der Bits 3 bis 6 aus EAX an die Position 0 bis 3 in EBX, alle anderen Bits in EBX sollen 0 sein

## 1.2 Programmverständnis

### 1.2.1 Statusregister

Kreuzen Sie die Status-Flags an, welche nach Ausführung eines jeden Befehls gesetzt sind. Jeder Befehl ist unabhängig zu betrachten; vor jedem Aufruf sind alle Status-Flags nicht gesetzt. Im Register EAX steht vor der Ausführung des Befehls der Wert „100“.

| Befehl | Carry | Zero | Sign | Overflow |
|---|---|---|---|---|
| sub al,200 | ____ | ____ | ____ | ____ |
| cmp al,100 | ____ | ____ | ____ | ____ |
| push ax | ____ | ____ | ____ | ____ |
| add al,100 | ____ | ____ | ____ | ____ |

### 1.2.2 Registerbelegung

Geben Sie die Inhalte der gefragten Register (in Hex oder Dec) nach der Ausführung des folgenden Programms an. Der Stapelspeicher ist zu Beginn leer (ESP=„4096“), die Werte der frei belegbaren Register alle „0“.

```
mov ax,7856h
mov cx,3412h
push eax
mov al,ch
mov ah,cl
pop cx
or dx,cx
shl ecx,8
mov bx,cx
ror edx,8
mov bl,dl
```

Registerwerte nach Programmausführung

AH:AL _____ BH:BL _____ CH:CL _____

DH:DL _____ ESP _____

## 1.3 Programmentwicklung

- Unterprogramme sollen neben den in der Aufgabenstellung explizit genannten keine weiteren Seiteneffekte haben.
- In der Aufgabenstellung nicht erwähnte Sonderfälle müssen bei der Bearbeitung nicht berücksichtigt werden.
- Zu nutzende Speicherbereiche dürfen als hinreichend groß und sich nicht überschneidend angenommen werden.

### 1.3.1 Mittelwert

Schreiben Sie ein per `CALL` aufrufbares Unterprogramm `mittelwert`, welches den ganzzahligen Mittelwert zweier in `EAX` und `EBX` übergebenen vorzeichenbehafteten Zahlen ohne die Verwendung von Divisionsbefehlen bildet und im Register `EAX` zurückgibt. Die Summe von `EAX` und `EBX` kann in 32 Bit dargestellt werden.

### 1.3.2 Formatumwandlung

- Eine C-Zeichenkette ist eine Folge unbestimmter Länge von Bytes (ASCII-Zeichen), welche durch ein `0`-Byte terminiert wird.
- Eine Pascal-Zeichenkette ist eine Folge von Bytes (ASCII-Zeichen), deren erstes Byte die Anzahl der anschließenden relevanten Bytes als vorzeichenlose Zahl angibt.

Schreiben Sie ein per `CALL` aufrufbares Unterprogramm `p2c`, welches eine Pascal-Zeichenkette an der in `EAX` gespeicherten Arbeitsspeicheradresse unmittelbar in eine C-Zeichenkette an die in `EBX` übergebenen Adresse konvertiert. Als Rückgabewert soll in `EAX` die Länge der Zeichenkette übergeben werden.

# Aufgabe 2 - Mikroprogrammierung

Sie haben ein Merkblatt mit einer Kurzbeschreibung der mikroprogrammierbaren Maschine erhalten, in welchem Sie alle zur Lösung der Aufgabe notwendigen Angaben finden, z.B. die Beschreibung des Mikroinstruktionsformats und die Funktionstabellen der Bausteine.

Das Mikroprogramm IFETCH, abgelegt ab Adresse 0x000 des Mikroprogrammspeichers, kann als gegeben betrachtet werden. Es holt den nächsten Maschinenbefehl aus dem Hauptspeicher in das Instruktionsregister, inkrementiert den Befehlszähler und springt das dem Befehls-Opcode zugehörige Mikroprogramm an. IFETCH benötigt 3 Takte zur Ausführung. Mikroprogramme müssen am Ende wieder zum Anfang des Mikroprogramms IFETCH zurückspringen. Hexadezimale Zahlen werden in Java- bzw. C-Schreibweise dargestellt und beginnen mit der Buchstabenkombination 0x, z.B: 0x1234, 0x011A oder 0xBEEF.

Gegeben sind folgende Maschinenbefehle:

| Opc. | Befehl | | Beschreibung |
|---|---|---|---|
| 0x11 | `MOV` | **RA**, **RB** | Kopiert den Inhalt von Register **RA** in das Register **RB**. |
| 0x20 | `INC` | **RB** | Inkrementiert **RB** um 1 (**RB** = **RB** + 1). Die Flags des Maschinenstatusregisters werden entsprechend des Ergebnisses gesetzt. |
| 0x29 | `AND` | **RB**, *imm* | Führt eine bitweise UND-Operation zwischen dem Inhalt von Register **RB** und *imm* durch. Das Ergebnis wird nach **RB** zurückgeschrieben. |
| 0x30 | `JZ` | *adr* | Bedingter Sprung nach *adr*, wenn das Zero-Flag im Maschinenstatusregister gesetzt ist. |
| 0x38 | `JMP` | *adr* | Unbedingter Sprung nach *adr*. |
| 0x40 | `ADD` | **RA**, **RB** | Addiert den Inhalt des Registers **RA** zum Inhalt von Register **RB**, und speichert das Ergebnis in Register **RB**. Die Flags des Maschinenstatusregisters werden entsprechend des Ergebnisses gesetzt. |
| 0x52 | `CMP` | [**RA**], *imm* | Führt eine Subtraktion des Inhalts der Speicherzelle, deren Adresse in Register **RA** steht, und des direkt gegebenen Wertes *imm* durch. Die Flags des Maschinenstatusregisters werden entsprechend des Ergebnisses gesetzt. Das Ergebnis wird verworfen. |
| 0x53 | `CMP` | [**RA**], **RB** | Führt eine Subtraktion des Inhalts der Speicherzelle, deren Adresse in Register **RA** steht, und des Wertes in Register **RB** durch. Die Flags des Maschinenstatusregisters werden entsprechend des Ergebnisses gesetzt. Das Ergebnis wird verworfen. |
| 0x61 | `CLRF` | | Alle Flags des Maschinenstatusregisters werden gelöscht. |
| 0x77 | `INTAB` | **RA**, **RB** | Überprüft, ob der Wert in Register **RB** in einer Hash-Tabelle im Hauptspeicher enthalten ist. Der Inhalt des Registers **RA** ist dabei die Startadresse der Hash-Tabelle. Wenn der Wert enthalten ist, wird das Z-Flag des Maschinenstatusregisters gesetzt, andernfalls gelöscht. Andere Flags sind danach undefiniert. Die Funktion von `INTAB` ist durch das in Aufgabe 3.2 gegebene Maschinenprogramm beschrieben (abgesehen von der Registernutzung). |

Anmerkung: Die Bezeichnung „**RA**“ (bzw. „**RB**“) steht hier abkürzend für „das durch das A-Registeradressfeld (bzw. B-Registeradressfeld) des Maschinenbefehlswortes adressierte Register“.

**2.1** Welche der oben angegebenen Befehle belegen

a) 16 Bit

---

b) 32 Bit

---

im Hauptspeicher? Bitte die jeweiligen Opcodes angeben!

**2.2** Im Folgenden sei eine Datenstruktur, die Mengen von 16-Bit-Werten repräsentiert, als Hash-Tabelle (mit sogenannter offener Adressierung) implementiert. Dazu gibt es eine Tabelle mit 256 Einträgen. Ein Wert $v$ ist genau dann enthalten, wenn $v$ im Tabelleneintrag an Position $v \bmod 256$ oder auf nachfolgenden Positionen (mit „Wrap-around“) gespeichert ist, solange kein Null-Eintrag existiert. Eine Null selbst kann nicht aufgenommen werden. Folgendes Maschinenprogramm überprüft, ob der Inhalt von Register r1 in der Datenstruktur enthalten ist. Die Basisadresse der dafür benutzten Tabelle steht in Register r0. Entsprechend des Ergebnisses der Prüfung wird das Z-Flag des Maschinenstatusregisters gesetzt.

```
                MOV   r1, r2
schleife:       AND   r2, 255
                MOV   r0, r3
                ADD   r2, r3
                CMP   [r3], 0
                JZ    nichtgefunden
                CMP   [r3], r1
                JZ    ende
                INC   r2
                JMP   schleife
nichtgefunden:  CLRF
ende:           ...
```

☐ **2.2.1** Ein Fehler in der gegebenen Implementierung ist das mögliche Auftreten einer Endlosschleife. Wann kann dies passieren? Welche Änderung verhindert das Problem?

☐ **2.2.2** Anstatt zweimal bei den CMP-Befehlen Hauptspeicherzugriffe durchzuführen, könnte man den Wert mit einem eigenen Befehl einmal aus dem Speicher laden und diesen zweimal vergleichen. Begründen Sie, ob diese Variante eine Verbesserung der Ausführungszeit darstellt. Geben Sie dabei an, von wievielen Takten Unterschied Sie in Bezug auf betroffene Maschinenbefehle ausgehen.

☐ **2.2.3** Um die Datenstruktur zu initialisieren, müssen 256 Werte in der Tabelle mit 0 beschrieben werden. Wie viele Takte benötigt eine MI-Implementierung in Form des Maschinenbefehls `CLEARTABLE RA, imm` (RA: Startadresse, imm: Länge) dafür mindestens?

☐ **2.2.4** Zeigen Sie, wie das gegebene Assemblerprogramm in hexadezimaler Codierung aussieht.

| Adresse | Inhalt | Befehl | | |
|---|---|---|---|---|
| 0x0100 | _____ | | `MOV` | r1, r2 |
| _____ | _____ | | | |
| _____ | _____ | schleife: | `AND` | r2, 255 |
| _____ | _____ | | | |
| _____ | _____ | | `MOV` | r0, r3 |
| _____ | _____ | | | |
| _____ | _____ | | `ADD` | r2, r3 |
| _____ | _____ | | | |
| _____ | _____ | | `CMP` | [r3], 0 |
| _____ | _____ | | | |
| _____ | _____ | | `JZ` | nichtgefunden |
| _____ | _____ | | | |
| _____ | _____ | | `CMP` | [r3], r1 |
| _____ | _____ | | | |
| _____ | _____ | | `JZ` | ende |
| _____ | _____ | | | |
| _____ | _____ | | `INC` | r2 |
| _____ | _____ | | | |
| _____ | _____ | | `JMP` | schleife |
| _____ | _____ | | | |
| _____ | _____ | nichtgefunden: | `CLRF` | |
| _____ | _____ | | | |
| _____ | | ende: | | |

**2.3** Folgende Maschinenbefehle sollen nun durch ein Mikroprogramm realisiert werden. Für `INTAB` sind die durchzuführenden Schritte in Kommentaren angegeben.

- `AND` **RB**, *imm*
- `CLRF`
- `INTAB` **RA**, **RB**

☐ **2.3.1** Um bei `CLRF` alle Flags des Maschinenstatusregisters zu löschen, soll in der ALU eine geeigneten Operation durchgeführt werden. Welche Operation schlagen Sie vor? Welche Felder einer Mikroinstruktion müssen dafür mit welchen Werten belegt werden?

---

---

---

☐ **2.3.2** Im ersten Takt von `INTAB` wird **RB** für spätere Modifikationen ins Q-Register gespeichert. Spricht etwas dagegen, dafür z.B. r15 zu benutzen? Wenn ja, warum?

---

---

☐ **2.3.3** Vervollständigen Sie die Tabellen auf den Seiten 18/19 (Ersatztabelle auf den Seiten 20/21).

Streichen Sie falsche Lösungen deutlich durch!
Bitte tragen Sie keine binären Werte ein, sondern verwenden Sie die Abkürzungen aus dem Merkblatt MI (in der Anlage).

Tabelle auf der nächsten Seite!

| | 79 | 78 | 77 | 76 | 75 | 74 | 73 | 72 | 71 | 70 | 69 | 68 | 67 | 66 | 65 | 64 | 63 | 62 | 61 | 60 | 59 | 58 | 57 | 56 | 55 | 54 | 53 | 52 | 51 | 50 | 49 | 48 | 47 | 46 | 45 | 44 | 43 | 42 | 41 | 40 | 39 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | IE | I3 | I2 | I1 | I0 | KMUX | K15 | K14 | K13 | K12 | K11 | K10 | K9 | K8 | K7 | K6 | K5 | K4 | K3 | K2 | K1 | K0 | I2 | I1 | I0 | I5 | I4 | I3 | I8 | I7 | I6 | A3 | A2 | A1 | A0 | ASEL | B3 | B2 | B1 | B0 | BSEL |
| | | Interrupt | | | | | Konstante | | | | | | | | | | | | | | | | Src | | | Func | | | Dest | | | RA Addr | | | | | RB Addr | | | | |

Adr. AND RB,imm

| | DIS | X | | X | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | DIS | X | | X | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| | DIS | X | | X | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |

CLRF

| | DIS | X | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|

INTAB RA,RB

RB nach Q

| | DIS | X | | X | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|

Q = Q AND 255

| | DIS | X | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|

RA + Q auf Adressbus

| | DIS | X | | X | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|

Datenbus nach r15, auf 0 überprüfen

| | DIS | X | | X | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|

falls nicht 0, nächste Zeile überspringen

| | DIS | X | | X | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|

nicht gefunden: Z-Flag (MA) löschen, nach IFETCH

| | DIS | X | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|

RB mit r15 vergleichen, Z-Flag (MA) entsprechend setzen

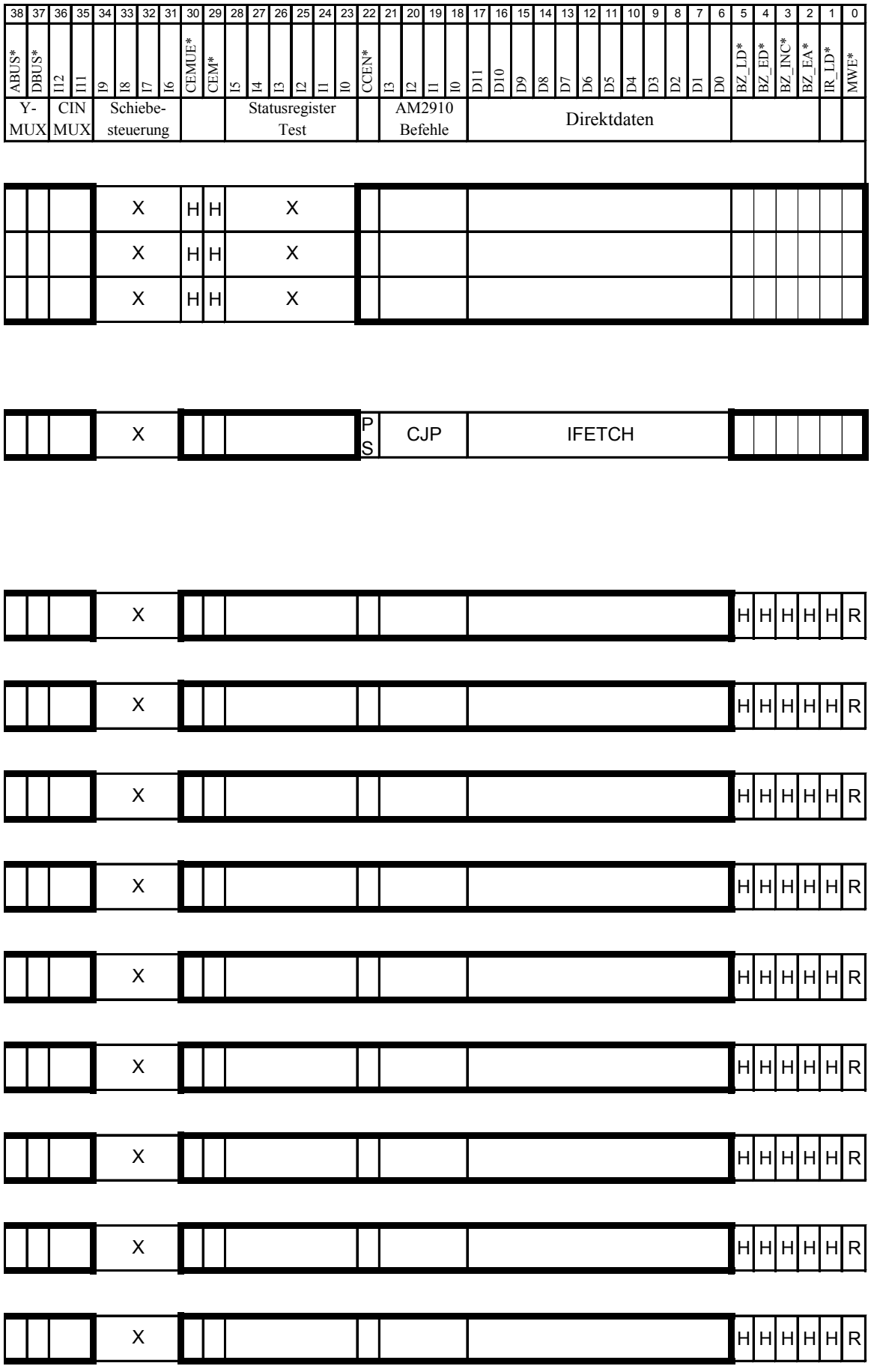
| | DIS | X | | X | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|

wenn gleich: gefunden, zurück nach IFETCH

| | DIS | X | | X | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|

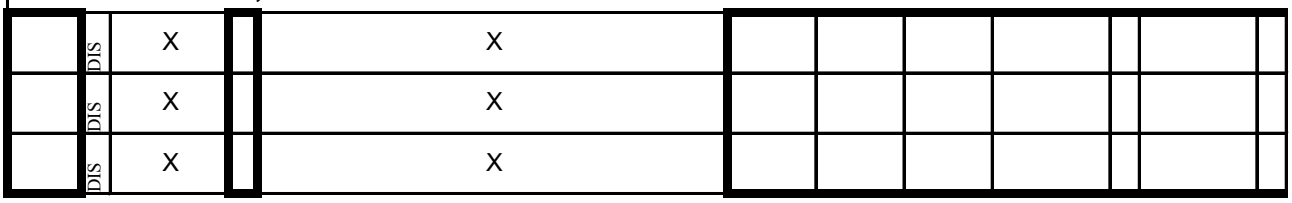
Q = Q +1, weitersuchen

| | DIS | X | | X | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|


38 37 36 35 34 33 32 31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
ABUS* DBUS* I12 I11 I9 I8 I7 I6 CEMUE* CEM* I5 I4 I3 I2 I1 I0 CCEN* I3 I2 I1 I0 D11 D10 D9 D8 D7 D6 D5 D4 D3 D2 D1 D0 BZ_LD* BZ_ED* BZ_INC* BZ_EA* IR_LD* MWE*
Y-MUX
CIN MUX
Schiebe-steuerung
Statusregister Test
AM2910 Befehle
Direktdaten
X H H X
X H H X
X H H X
X
PS
CJP
IFETCH
X H H H H H R
X H H H H H R
X H H H H H R
X H H H H H R
X H H H H H R
X H H H H H R
X H H H H H R
X H H H H H R
X H H H H H R

| | 79 | 78 | 77 | 76 | 75 | 74 | 73 | 72 | 71 | 70 | 69 | 68 | 67 | 66 | 65 | 64 | 63 | 62 | 61 | 60 | 59 | 58 | 57 | 56 | 55 | 54 | 53 | 52 | 51 | 50 | 49 | 48 | 47 | 46 | 45 | 44 | 43 | 42 | 41 | 40 | 39 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | IE | I3 | I2 | I1 | I0 | KMUX | K15 | K14 | K13 | K12 | K11 | K10 | K9 | K8 | K7 | K6 | K5 | K4 | K3 | K2 | K1 | K0 | I2 | I1 | I0 | I5 | I4 | I3 | I8 | I7 | I6 | A3 | A2 | A1 | A0 | ASEL | B3 | B2 | B1 | B0 | BSEL |
| | | Interrupt | | | | | Konstante | | | | | | | | | | | | | | | | Src | | | Func | | | Dest | | | RA Addr | | | | | RB Addr | | | | |

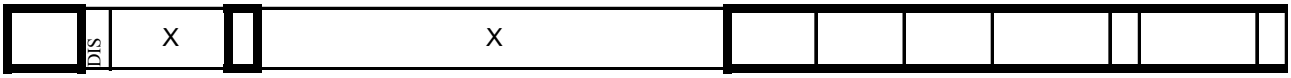
Adr. AND RB,imm




CLRF

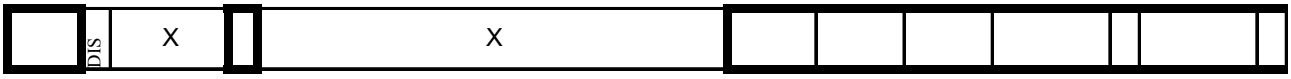

INTAB RA,RB

RB nach Q

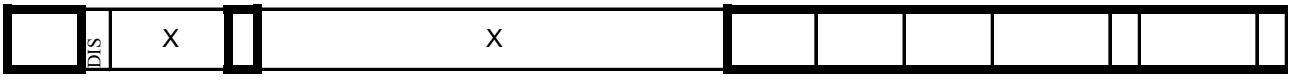



Q = Q AND 255

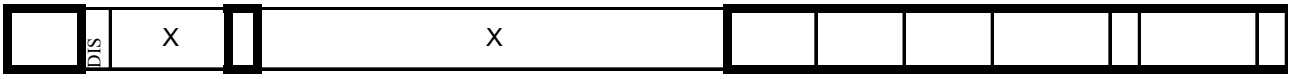

RA + Q auf Adressbus




Datenbus nach r15, auf 0 überprüfen

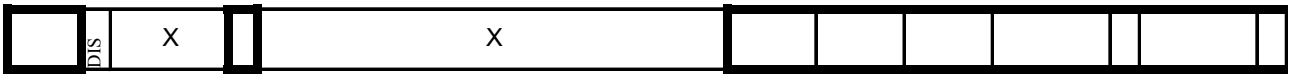



falls nicht 0, nächste Zeile überspringen

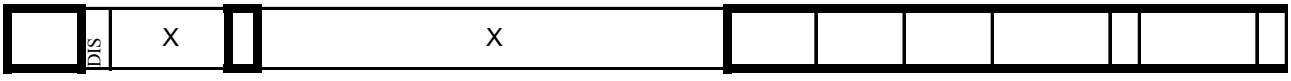



nicht gefunden: Z-Flag (MA) löschen, nach IFETCH

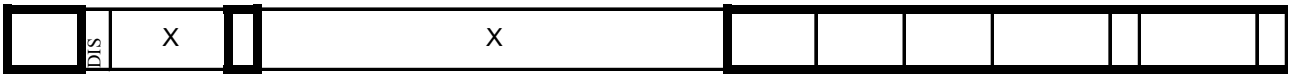

RB mit r15 vergleichen, Z-Flag (MA) entsprechend setzen

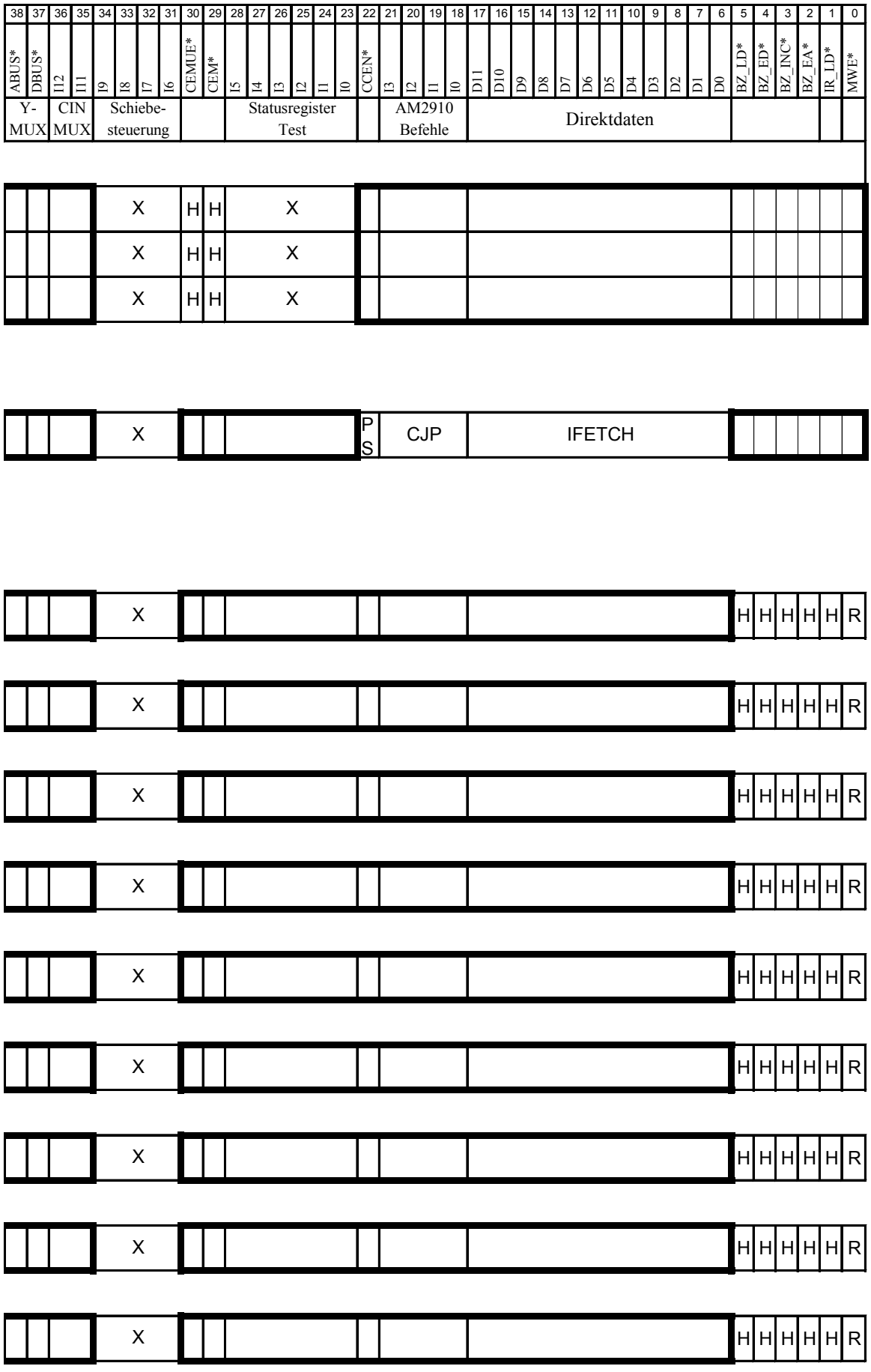



wenn gleich: gefunden, zurück nach IFETCH




Q = Q +1, weitersuchen

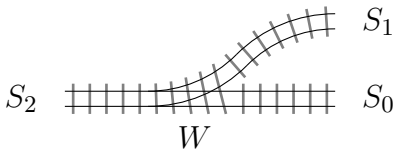



38 37 36 35 34 33 32 31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
ABUS* DBUS* I12 I11 I9 I8 I7 I6 CEMUE* CEM* I5 I4 I3 I2 I1 I0 CCEN* I3 I2 I1 I0 D11 D10 D9 D8 D7 D6 D5 D4 D3 D2 D1 D0 BZ_LD* BZ_ED* BZ_INC* BZ_EA* IR_LD* MWE*
Y-MUX
CIN MUX
Schiebe-steuerung
Statusregister Test
AM2910 Befehle
Direktdaten
X H H X
X H H X
X H H X
X
PS
CJP
IFETCH
X H H H H H R
X H H H H H R
X H H H H H R
X H H H H H R
X H H H H H R
X H H H H H R
X H H H H H R
X H H H H H R

# Aufgabe 3 - Rechnergestützter Schaltungsentwurf

## 3.1 Modelleisenbahnweiche

Eine Weiche einer Modelleisenbahn verbindet drei Gleissegmente $S_0$, $S_1$ und $S_2$. Fährt ein Zug „stumpf“, das heißt über $S_0$ oder $S_1$, in die Weiche ein, verlässt er sie immer über $S_2$. Wird die Weiche jedoch „spitz“ über $S_2$ befahren, lenkt sie den Zug geradeaus über $S_0$, falls das Kontrollsignal $w$ den Wert $0$ besitzt. Ansonsten wird der Zug auf $S_1$ gelenkt.




| | $a_1$ | $a_0$ |
|---|---|---|
| $S_0$ | 0 | 0 |
| $S_1$ | 0 | 1 |
| $S_2$ | 1 | 0 |

Es soll eine Schaltung entwickelt werden, die vorhersagt, auf welchem Gleissegment ein einkommender Zug die Weiche verlassen wird. Als Eingänge stehen das Kontrollsignal $w$ sowie eine kodierte Fassung $a_1a_0$ des Gleissegments, über das die Weiche befahren wird, zur Verfügung (siehe Tabelle). Die Ausgänge $b_1$ und $b_0$ repräsentieren das Gleissegment, über das die Weiche verlassen wird. Die Kodierung entspricht der von $a_1$ und $a_0$.

☐ **3.1.1** Ergänzen Sie die Wertetabelle der Schaltung!

| $a_1$ | $a_0$ | $w$ | $b_1$ | $b_0$ |
|---|---|---|---|---|
| 0 | 0 | 0 | | |
| 0 | 0 | 1 | | |
| 0 | 1 | 0 | | |
| 0 | 1 | 1 | | |
| 1 | 0 | 0 | | |
| 1 | 0 | 1 | | |

☐ **3.1.2** Wieso sind bei dieser Wertetabelle 6 Zeilen ausreichend? Wie viele Zeilen sind im Allgemeinen für die Wertetabelle einer Funktion mit 3 Eingängen erforderlich?

☐ **3.1.3** Geben Sie eine konjunktive Normalform für $b_0$ an!

☐ **3.1.4** Geben Sie eine disjunktive Normalform für $b_1$ an!

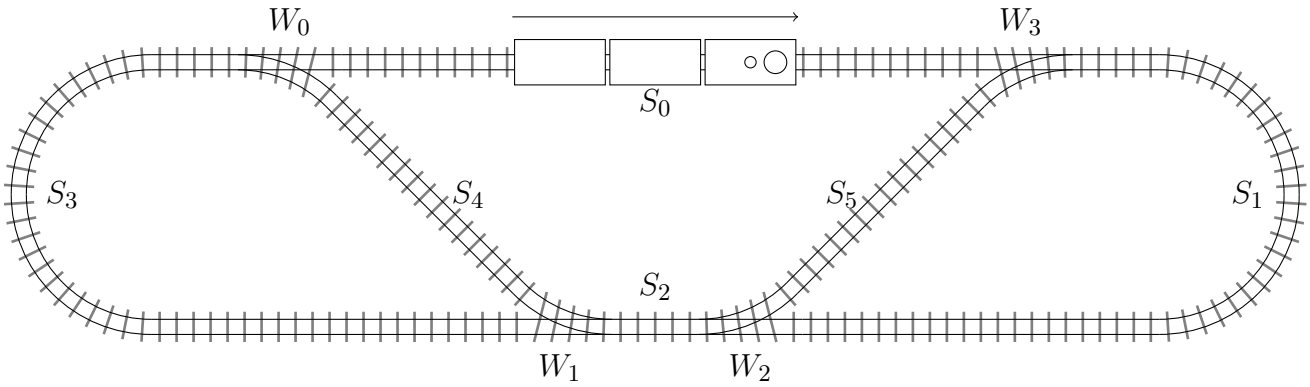
☐ **3.1.5** Erstellen Sie eine VHDL-Architecture, die die Schaltung beschreibt!

Die folgende VHDL-Entity-Deklaration ist gegeben:

```
entity weiche is
    port(
        a: in  unsigned(1 downto 0);
        w: in  std_logic;
        b: out unsigned(1 downto 0)
    );
end entity;
```

## 3.2 Modelleisenbahn

Nun wird ein Aufbau aus 4 Weichen $W_0$, $W_1$, $W_2$ und $W_3$, 6 Gleissegmenten $S_0$, $S_1$, $S_2$, $S_3$, $S_4$ und $S_5$ sowie einem Zug betrachtet.




Ein endlicher Zustandsautomat soll die aktuelle Position des Zuges vorhersagen und ist wie folgt spezifiziert:

- Ein asynchroner Reseteingang `rst` teilt dem Automaten mit, dass der Zug sich gerade auf Segment $S_0$ in der gezeigten Richtung bewegt.
- Dem Automat stehen über den Eingang `w` (4-bit-Vektor) die Kontrollsignale der Weichen zur Verfügung.
- Liegt bei einer steigenden Taktflanke von `clk` am bool'schen Eingang `trigger` eine $1$ an, so hat der Zug in diesem Moment eine der Weichen passiert. Der Zug passiert maximal eine Weiche pro steigender Taktflanke.
- Am Ausgang `b` (3-bit-Vektor) soll stets der Index des aktuell befahrenen Segments in binärer Kodierung ausgegeben werden.

☐ **3.2.1** Erstellen Sie die VHDL-Entity des Bausteins! Halten Sie sich an die vorgegebenen Signalnamen.

**3.2.2** Wieviele interne Zustände hat der Automat mindestens?

**3.2.3** Zeichnen Sie das Zustandsübergangsdiagramm („Blasendiagramm“) des Bausteins!

*Hinweis: Zeichnen Sie zuerst die beiden Teilautomaten, in denen der Zug auf den äußeren Schienen im Kreis fährt, also ohne Betrachtung von $S_4$ und $S_5$, und vervollständigen Sie das Bild anschließend zum vollständigen Automaten!*

☐ **3.2.4** Erstellen Sie die VHDL-Architecture des Bausteins!

*Hinweis: Es bietet sich an, im* `process` *als Standardwert für den neuen Zustand zuerst davon auszugehen, dass der Zug auf den äußeren Schienen fährt und dort bleibt. Anschließend lassen sich die Bedingungen für die Weichen überprüfen und der Standardwert gegebenenfalls überschreiben.*

*Zusätzlicher Platz für Lösungen - Bitte immer Aufgabennummer angeben!*

Lösung für Aufgabe . . .

Lösung für Aufgabe ...

Lösung für Aufgabe ...

Lösung für Aufgabe ...

Lösung für Aufgabe ...