Aufgabe 1)
Die folgenden Umwandlungen/Berechnungen beziehen sich auf das 32-Bit Single-Precision Format nach IEEE-754.

a) Stellen Sie die Zahl 7,625 in folgender Tabelle dar!

b) Wie werden denormalisierte Zahlen abgespeichert?

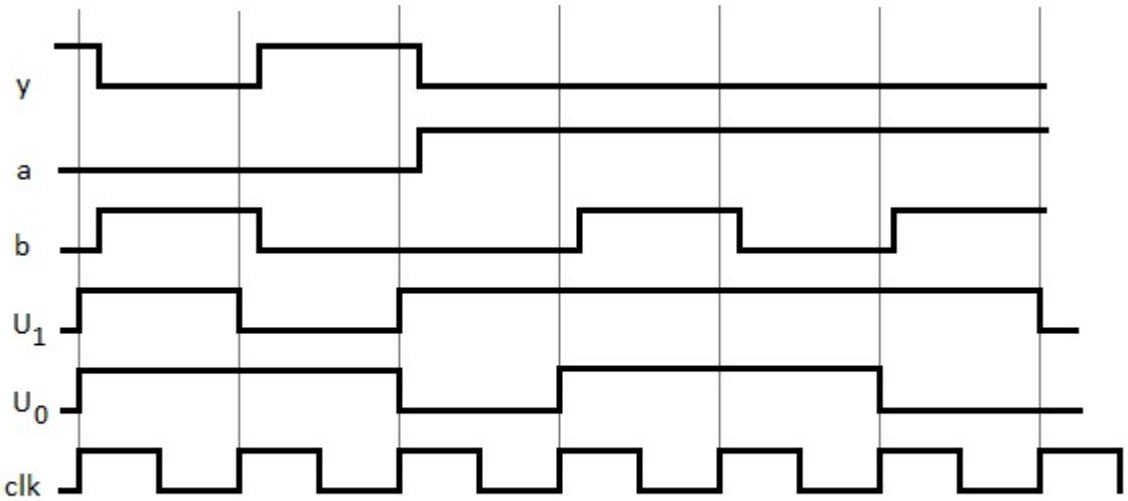
c) Welchen Wert hat die kleinste darstellbare denormalisierte Zahl?

d) Führen Sie für die gegebenen Zahlen eine Addition durch.

1	1	0	0	0	0	0	1	1	0	0	1	0	1	1	0	0	1	1	1	0	1	1	0	0	0	1	0	0	0	0	0
0	0	1	1	1	1	1	1	0	1	0	0	1	1	1	0	1	1	0	0	1	1	0	0	1	1	0	0	0	0	0	0

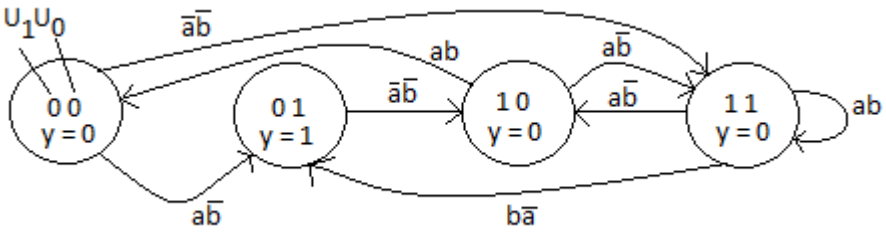
Aufgabe 2)

Gegeben sei das folgende Impulsdiagramm, mit dem Eingängen *a* und *b* und dem Ausgang *y* eines Synchronschaltwerkes. Die Gatterlaufzeiten seien willkürlich gewählt und spielen hier keine entscheidende Rolle.

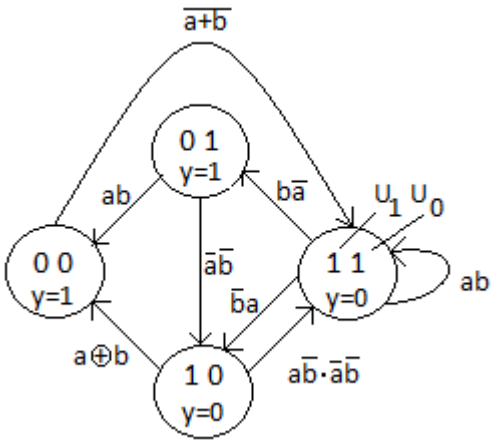


a) Entscheiden Sie, welche Zustandsdiagramme zu dem gegebenen Impulsdiagramm passen und welche nicht.

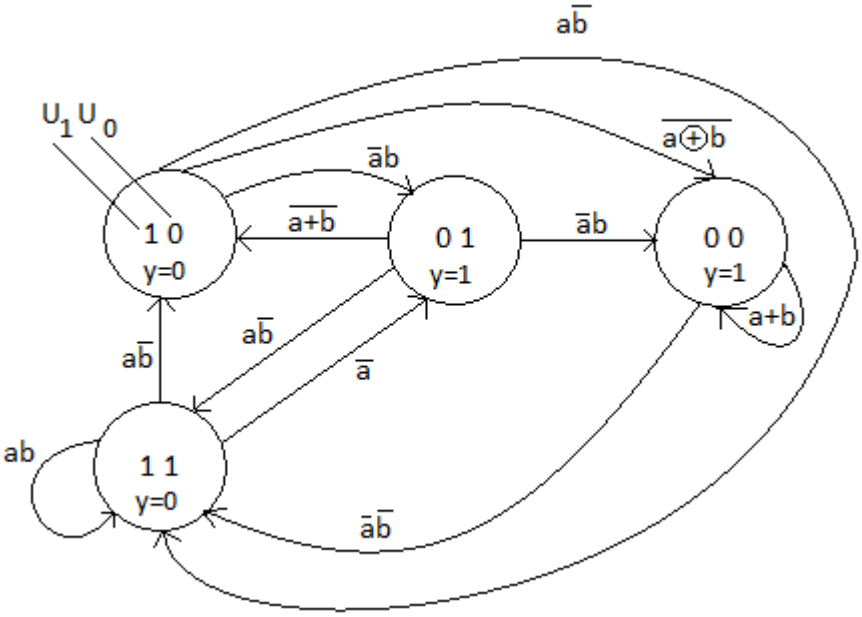
a)



b)



c)



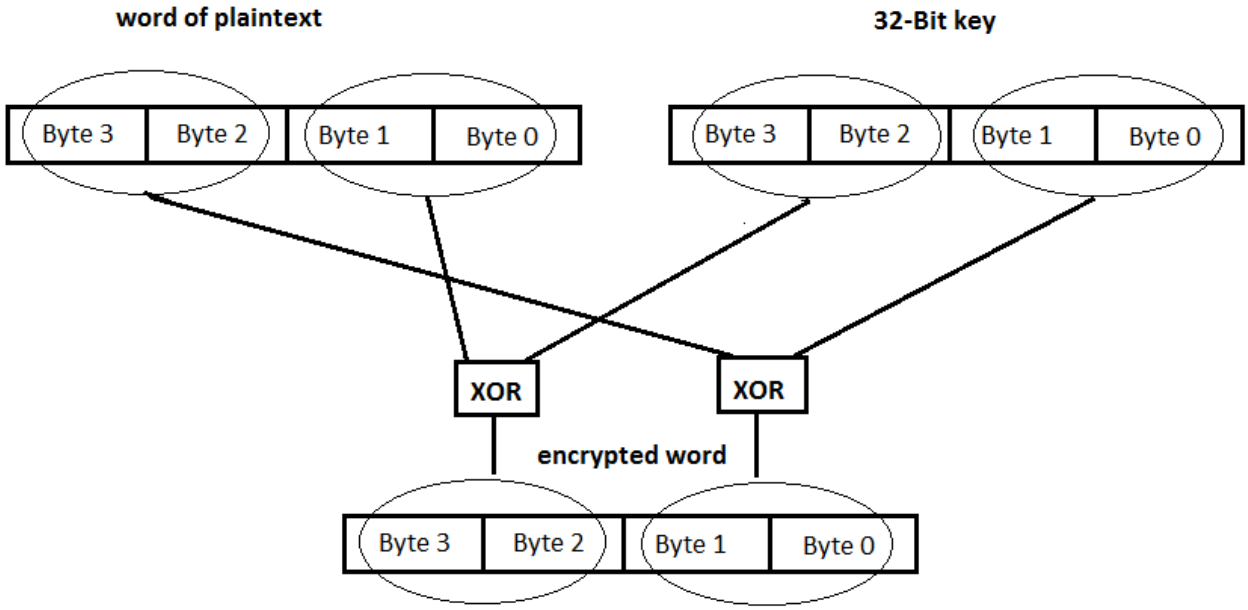
b) Betrachten Sie das Zustandsdiagramm 1.1.c). Bestimmen Sie die minimalen Übergangsfunktionen für dieses System!

c) Bestimmen Sie die Beschaltungsfunktionen der Flipflops. Benutzen Sie zum Speichern für das höherwertige Bit der Zustandscodierung ein D-Flipflop, für das niederwertige Bit ein JK-Flipflop.

d) Bestimmen Sie die disjunktive Normalform für das Ausgangssignal y.

e) Zeichnen Sie die Schaltung.

Aufgabe 3)
In der folgenden Aufgabe soll ein besonderer Verschlüsselungsalgorithmus in MIPS Assembler programmiert werden. Bei dieser Verschlüsselungsmethode wird mithilfe eines gegebenen 32 Bit Schlüssels jedes Element (ein Wort) eines beliebiges 32 Bit Integer-Arrays auf folgende Art und Weise verschlüsselt:



Das Prinzip soll an einem folgenden Beispiel nochmal verdeutlicht werden. Sie haben ein zu verschlüsselndes 32 Bit-Wort-Array namens „arrayPlaintext“ und einen 32 Bit Schlüssel mit dem Wert 0xFFFFFFFF. Die verschlüsselten Werte werden in einem Zielarray „arrayEncoded“ abgelegt.

```
arrayPlaintext:     .word  0x000000001 0x000000FF 0xFF000002
key:                .word  0xFFFFFFFF
```

Für dieses konkrete Beispiel würde sich somit ergeben:
(Es gilt: a xor 1 = $\bar{a}$):

```
arrayEncoded:       .word  0xFFFEFFFF  0xFF00FFFF 0xFFFD00FF
```

Programmieren Sie in MIPS Assembler eine Funktion „encode“, die ein beliebiges Array mit einem beliebigen 32-Bit Schlüssel kodiert. Im Parameterregister $a0 wird dabei die Basisadresse des zu verschlüsselnden Arrays übergeben. Im Parameterregister $a1 steht die Länge des zu verschlüsselnden Arrays und in Parameterregister $a2 wird die Basisadresse des Zielarrays übergeben, in dem die verschlüsselten Werte abgelegt werden sollen. Im Register $a3 wird der 32-Bit Schlüssel übergeben. Kommentieren Sie jede Zeile!

Aufgabe 4)
Beziehen Sie sich in den folgenden Aufgaben die Abarbeitung von MIPS-Assembler-Befehlen auf dem MIPS Pipelinedatenpfad wie in Abbildung 3 dar gestellt..

a) Betrachten Sie die Abarbeitung folgender Befehlssequenz auf dem Pipelinedatenpfad:

```
add   $t0,$t1,$s2
lw    $t3,4($t1)
and   $t2,$t3,$t0
ori   $t9,$t1,$t3
sw    $t9,-4($sp)
```

Bestimmen Sie für das obige Programmstück taktweise die Ausgangssignale der Forwarding-Einheit, mit denen die Multiplexer vor den ALU-Eingängen angesteuert werden.

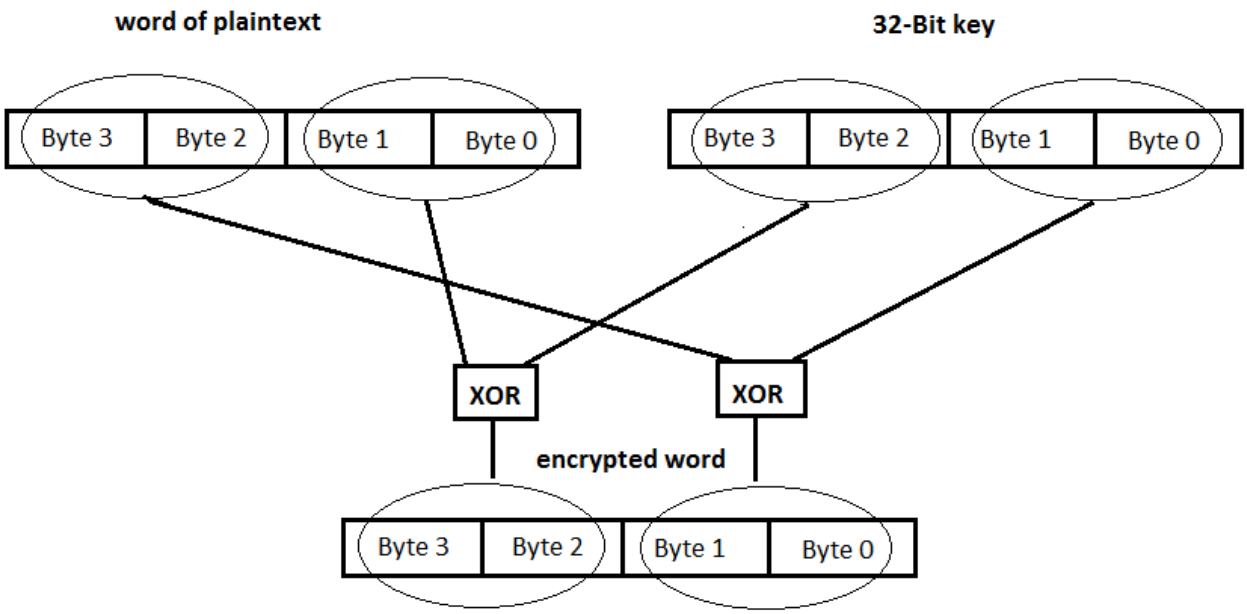
	ForwardA	ForwardB
Takt 1	**	**
Takt 2	**	**
Takt 3		
Takt 4		
Takt 5		
Takt 6		
Takt 7		
Takt 8		

b) Was versteht man unter einem Load-Use-Konflikt?

c) Tritt ein solcher Konflikt im obigen Programmfragment auf?

d) Wie wird er allgemein aufgelöst?

e) Betrachten Sie wieder den folgenden Verschlüsselungsalgorithmus für 32 Bit Wörter.

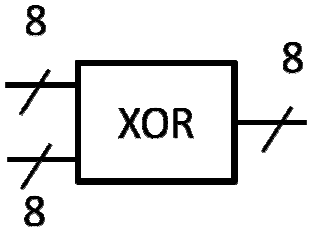


Ein Beispiel:
Das Wort „0xFF000002“ würde mit dem Schlüssel „0xFFFFFFFF“ demnach zu „0xFFFD00FF“ verschlüsselt werden.

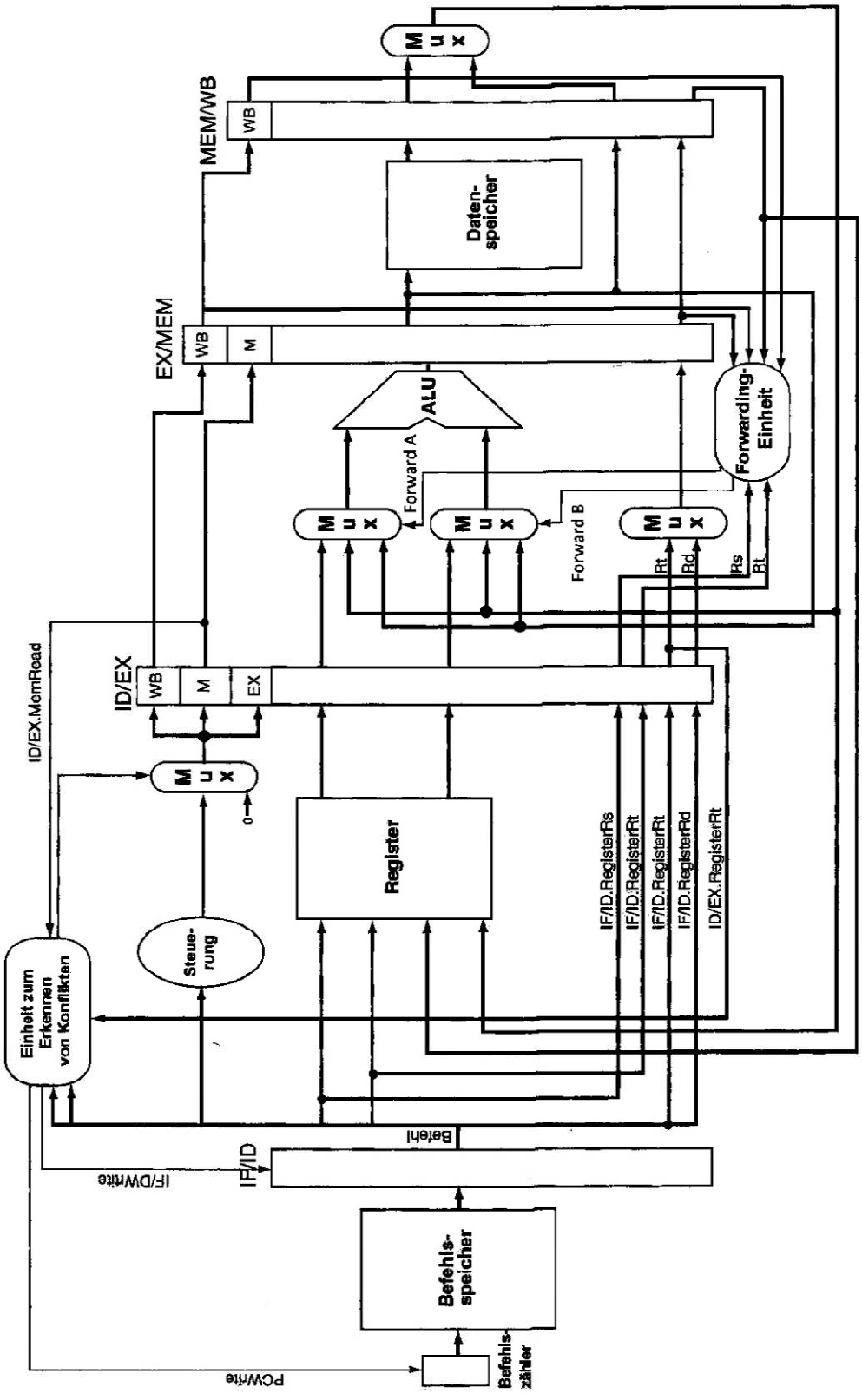
Für die Entschlüsselung soll der Pipelinedatenpfad nun um den neuen Befehl „dec r1,r2,r3“ erweitert werden.

In Register r2 steht dabei das verschlüsselte 32-Bit Wort und in Register r3 der 32-Bit Schlüssel, der für die Entschlüsselung benötigt wird. Das entschlüsselte Klartextwort soll in Register r1 geschrieben werden.

Die ALU beherrscht keine XOR-Operation. Sie haben ausschließlich mehrere 8-Bit XOR Bausteine zur Verfügung



Erweitern Sie den Datenpfad derart, dass der Entschlüsselungsbefehl von der Hardware unterstützt wird. Fügen Sie gegebenenfalls auch neue Steuerleitungen und Multiplexer ein.


M u x
MEM/WB
WB
Daten-speicher
EX/MEM
WB
M
ALU
Forward A
Forward B
Forwarding-Einheit
M u x
M u x
M u x
Rt
Rd
Rs
Rt
ID/EX
WB
M
EX
ID/EX.MemRead
M u x
0
Register
Steue-rung
Einheit zum Erkennen von Konflikten
IF/ID.RegisterRs
IF/ID.RegisterRt
IF/ID.RegisterRt
IF/ID.RegisterRd
ID/EX.RegisterRt
Befehl
IF/ID
IFIDWrite
Befehls-speicher
PCWrite
Befehls-zähler