UNIVERSITÄT LEIPZIG

Grundlagen der Technischen Informatik 2
Sommersemester 25
Übungsblatt 4

Reduced Instruction Scente

Aufgabe 1: Rechnerarchitektur

1. Was sind Unterschiede zwischen RISC und CISC Architekturen. Nennen Sie je ein Beispiel einer RISC bzw. CISC Architektur.
2. Was sind Unterschiede zwischen Harvard und von Neumann Architekturen. Nennen Sie je ein Beispiel.

Aufgabe 2: Halbleiterpeicher

1. Ordnen Sie die Speicherarten dem beschriebenen Verhalten in der Tabelle zu.
   Speicherarten: NV-RAM, ROM, SRAM, EPROM, PROM, DRAM

Speicherart	Programmierbar	Reversibel	Schreibbar	Statisch	Flüchtig
				X	
	X			X	
	X	X		X	
	X	X	X	X	X
	X	X	X		X
	X	X	X	X	

Aufgabe 3: Program Counter

In dieser Aufgabe soll ein simpler 4-Bit Programm Counter entworfen werden.

1. Machen Sie sich mit einem Programm Counter vertraut. Was unterscheidet diesen von einem regulären Counter? Welche zusätzliche Funktionalität muss dieser haben, um in dem Programm direkte Sprünge auszuführen?
2. Entwerfen Sie einen synchronen 4-Bit Zähler.
3. Erweitern Sie Ihren Zähler, um spezifische Adressen speichern zu können.
   Dafür erhält der Counter zwei zusätzliche Eingaben: Steuersignal s_0, eine 4-Bit Adresse $a_0 ... a_3$.
   Der Programm Counter soll sich wie folgt verhalten:

Steuersignal	4-Bit Adresse	Vorzustand	Neuer Zustand	Funktion
s_0	$a_0 ... a_3$	$q_{0_{-1}} ... q_{3_{-1}}$	$q_0 ... q_3$	
0	Adr	$\mathtt{Q}_{-1}$	$\mathtt{Q} = \mathtt{Q}_{-1} + 1$	Zählen
1	Adr	$\mathtt{Q}_{-1}$	$\mathtt{Q} = \mathtt{Adr}$	Springen

Aufgabe 1: Rechnerarchitektur

1. Was sind Unterschiede zwischen RISC und CISC Architekturen. Nennen Sie je ein Beispiel einer RISC bzw. CISC Architektur.
2. Was sind Unterschiede zwischen Harvard und von Neumann Architekturen. Nennen Sie je ein Beispiel.

1.

RISC = Reduced Instruction . . .

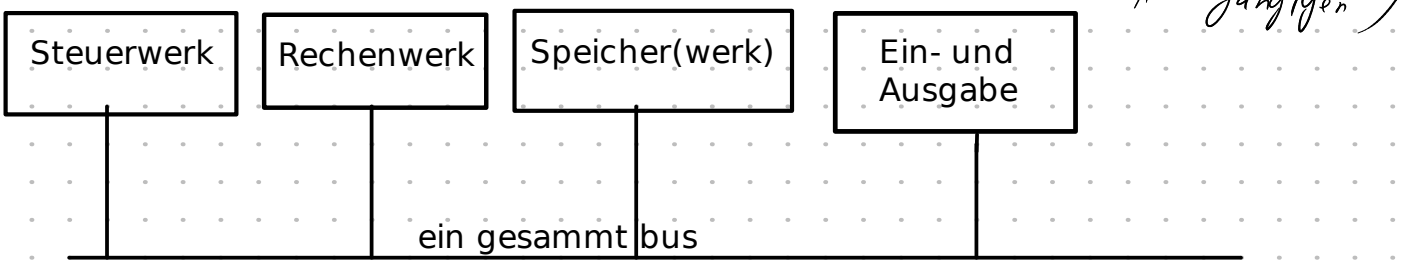
weniger Befehle (ARM)
schneller (2i-Mini Rechn.)

CISC = Complex
(x86) (2a-Mini Rechner)

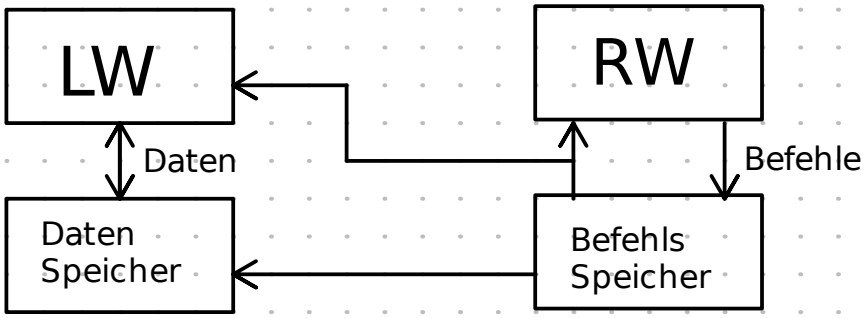
2 Neumann Progamm und Datenspeicher getrennt

weniger Hardware Komponente

(2i-Mini Rechner
Alle gängigen)



Harvard



Aufgabe 2: Halbleiterpeicher

1. Ordnen Sie die Speicherarten dem beschriebenen Verhalten in der Tabelle zu.
   Speicherarten: NV-RAM, ROM, SRAM, EPROM, PROM, DRAM
   static ram, mit flipflops
   dynamic, kondensatoren

Speicherart	Programmierbar	Reversibel	Schreibbar	Statisch	Flüchtig	
ROM				X		BIOS odere UEvi
PROM	X			X		Micro controller
EPROM	X	X		X		Firmenspeicher
SRAM	X	X	X	X	X	CPU Cache
DRAM	X	X	X		X	Hauptspeicher
NV-RAM	X	X	X	X		konfig von bios

Programmierbar: kann beschrieben werden
Reversible: kann neu beschrieben werden nachdem alles gelöscht wurde
Schreibbar: kann alles überschreiben
Statisch: hält daten solange strom vorhanden ist,
ohne das daten aufgefrisch werden müssen
Flüchtig: muss daten auffrischen damit behalten werden

Aufgabe 3: Program Counter

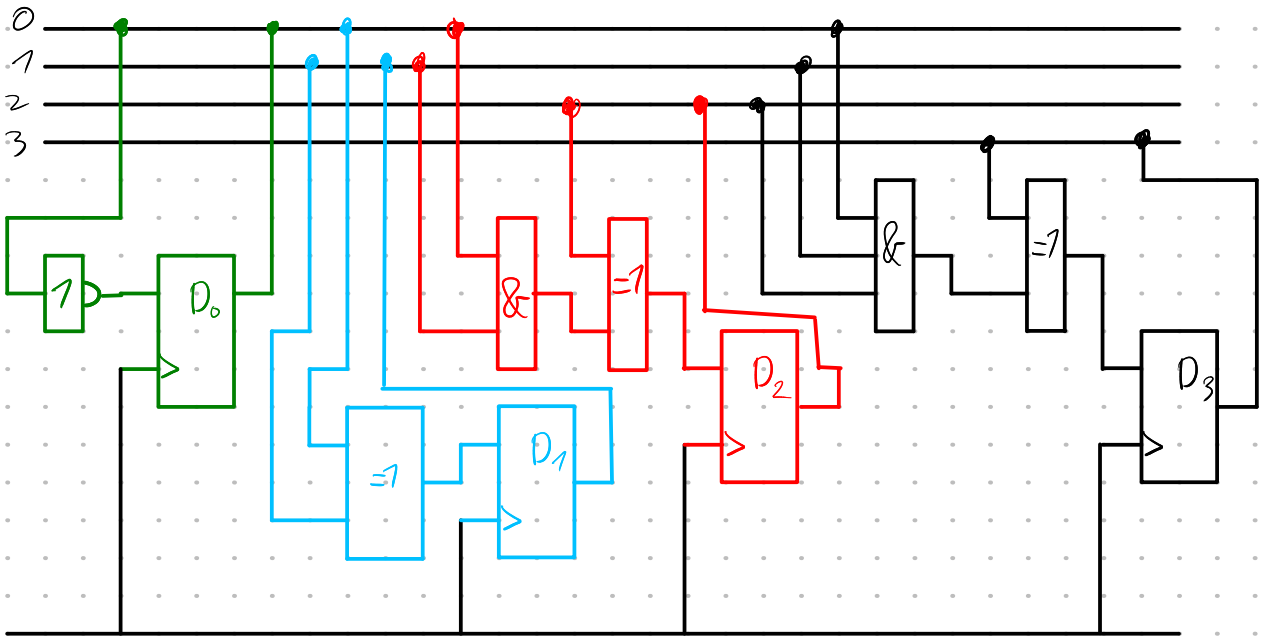
In dieser Aufgabe soll ein simpler 4-Bit Programm Counter entworfen werden.

1. Machen Sie sich mit einem Programm Counter vertraut. Was unterscheidet diesen von einem regulären Counter? Welche zusätzliche Funktionalität muss dieser haben, um in dem Programm direkte Sprünge auszuführen?
2. Entwerfen Sie einen synchronen 4-Bit Zähler.
3. Erweitern Sie Ihren Zähler, um spezifische Adressen speichern zu können.
   Dafür erhält der Counter zwei zusätzliche Eingaben: Steuersignal s_0, eine 4-Bit Adresse $a_0 ... a_3$.
   Der Programm Counter soll sich wie folgt verhalten:

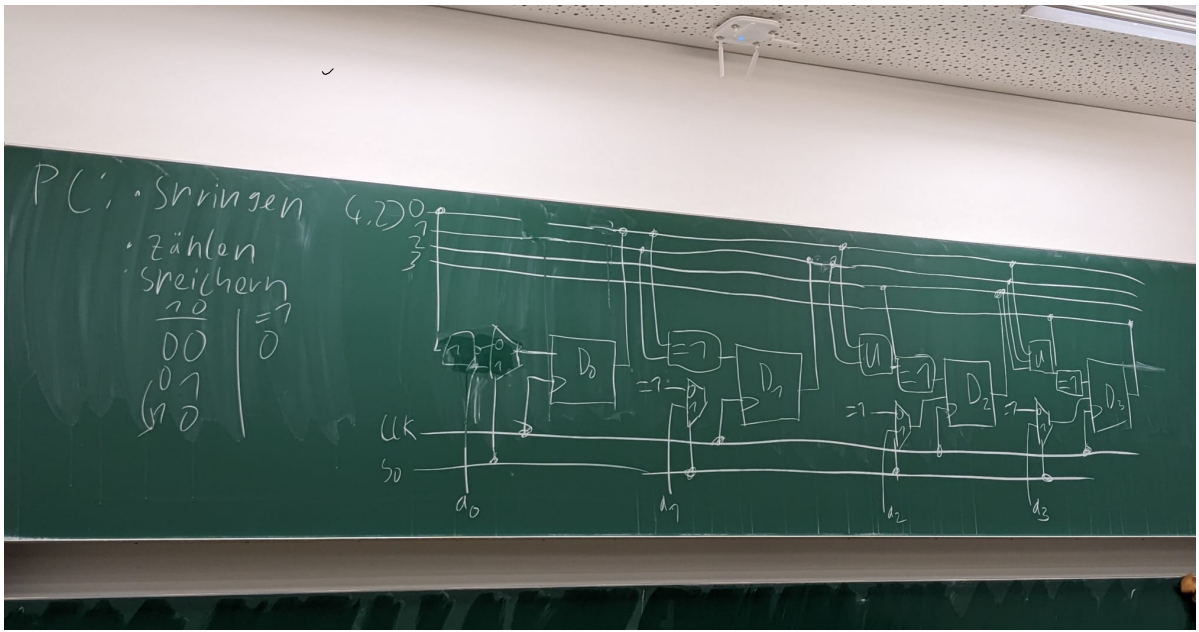
Steuersignal	4-Bit Adresse	Vorzustand	Neuer Zustand	Funktion
s_0	$a_0 ... a_3$	$q_{0_{-1}} ... q_{3_{-1}}$	$q_0 ... q_3$	
0	Adr	Q_{-1}	$Q = Q_{-1} + 1$	Zählen
1	Adr	Q_{-1}	$Q = Adr$	Springen

1. muss es erlauben an eine bestimmte Stelle zu springen
   zählen
   speichern der aktuellen addresse

2.



3.



Aufgabe 4: Mini ALU

Sei ein Schaltwerk mit zwei Inputs A und B, zwei Speicherregistern R_0 und R_1, zwei MUX-Steuersignalen m_0 und m_1, einem DEMUX-Steuersignal s_0 und einem Takt `CLK` gegeben.
Konstruieren Sie ein Schaltwerk, welches abhängig von den Steuersignalen entweder A, B, $A + B$ oder $A - B$ in eines der Speicherregister speichert.
Hinweis: Bekannte oder bereits konstruierte Schaltnetze wie Adder, Subtracter und D-FlipFlop müssen nicht aus Gattern konstruiert werden.

1. Konstruieren Sie zunächst einen MUX, dessen 4 Inputs in Abhängigkeit der Steuersignale die folgenden Funktionen darstellen.

m_0	m_1	Funktion
0	0	A
0	1	B
1	0	$A + B$
1	1	$A - B$

2. Konstruieren Sie einen DEMUX, der seinen Input, abhängig vom Steuersignal an eins der Speicherregister weiterleitet.

s_0	Register
0	R_0
1	R_1

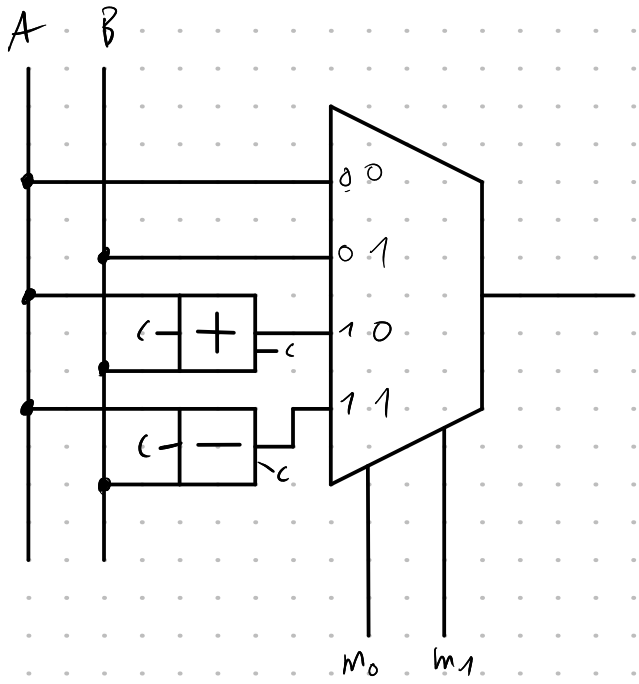
3. Verbinden Sie den Takt mit den Clock-Inputs der Register so, dass ein Register nur überschrieben wird, wenn es auch mit dem DEMUX ausgewählt ist.
4. Verbinden Sie nun den Output des MUX mit dem Input des DEMUX und verwenden Sie die Q-Outputs der Register als A und B.
5. Welche Operationen kann diese simple ALU durchführen?
6. Wie lässt sich diese ALU auf n Bit erweitern?

Aufgabe 4: Mini ALU

Sei ein Schaltwerk mit zwei Inputs A und B, zwei Speicherregistern R_0 und R_1, zwei MUX-Steuersignalen m_0 und m_1, einem DEMUX-Steuersignal s_0 und einem Takt `CLK` gegeben.
Konstruieren Sie ein Schaltwerk, welches abhängig von den Steuersignalen entweder A, B, $A + B$ oder $A - B$ in eines der Speicherregister speichert.
Hinweis: Bekannte oder bereits konstruierte Schaltnetze wie Adder, Subtracter und D-FlipFlop müssen nicht aus Gattern konstruiert werden.

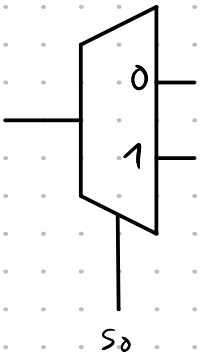
1. Konstruieren Sie zunächst einen MUX, dessen 4 Inputs in Abhängigkeit der Steuersignale die folgenden Funktionen darstellen.

m_0	m_1	Funktion
0	0	A
0	1	B
1	0	$A+B$
1	1	$A-B$

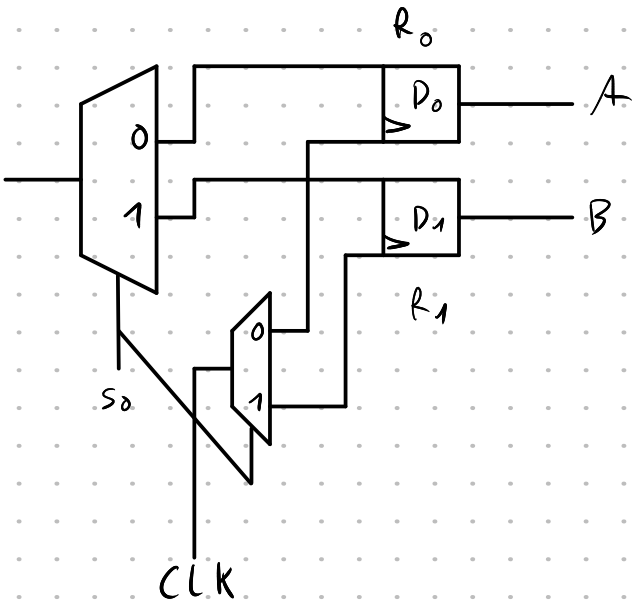


2. Konstruieren Sie einen DEMUX, der seinen Input, abhängig vom Steuersignal an eins der Speicherregister weiterleitet.

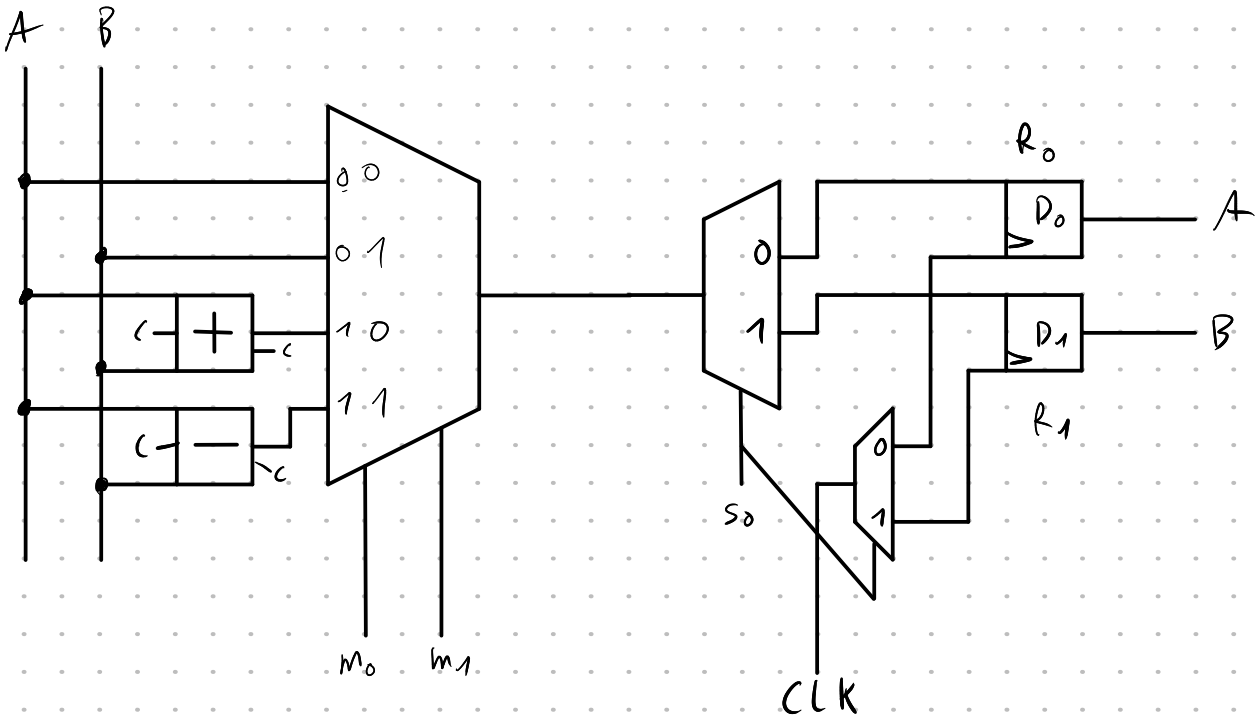
s_0	Register
0	R_0
1	R_1



3. Verbinden Sie den Takt mit den Clock-Inputs der Register so, dass ein Register nur überschrieben wird, wenn es auch mit dem DEMUX ausgewählt ist.



4. Verbinden Sie nun den Output des MUX mit dem Input des DEMUX und verwenden Sie die Q-Outputs der Register als A und B.



5. Welche Operationen kann diese simple ALU durchführen?

m_0	m_1	s_0	F
0	0	0	$R_0 = R_0$
0	0	1	$R_1 = R_0$
0	1	0	$R_0 = R_1$
0	1	1	$R_1 = R_1$
1	0	0	$R_0 = R_0 + R_1$
1	0	1	$R_1 = R_0 + R_1$
1	1	0	$R_0 = R_0 - R_1$
1	1	1	$R_0 = R_1 - R_0$

6. Wie lässt sich diese ALU auf n Bit erweitern?