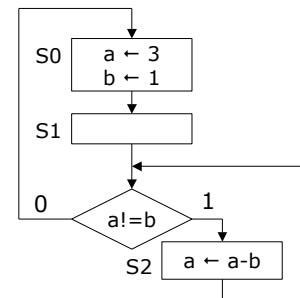


Schlüssel: 0-20 Punkte: Nicht Genügend; 21-25 Punkte: Genügend; 26-30 Punkte: Befriedigend;
31-35 Punkte: Gut; 36-40 Punkte: Sehr Gut.

1. Gegeben sind zwei 4-Bit-Register A und B mit den Inhalten A = 0x3, B = 0x9 (hexadezimal notiert). Zeigen Sie eine Schaltung, mit deren Hilfe man die Summe A+B bildet und in C speichert. Zur Verfügung stehen Volladdierer und Flipflops. Interpretieren Sie den Inhalt von C unter der Annahme, dass es sich um vorzeichenlose (unsigned) Binärzahlen handelt. Wie sieht das aus, wenn es sich um vorzeichenbehaftete (signed) Zahlen handelt? Erläutern Sie Ihre Antwort. (10 Punkte)

2. Zeigen Sie für die nebenstehende ASM ein Zeitdiagramm für die ersten 10 Taktperioden (Anfangszustand ist S0). Erläutern Sie Ihre Antwort. Entwerfen Sie den always-Block eines Verilog-Modells dieser ASM in „Implicit Style Coding“. (10 Punkte)



3. Was versteht man im Zusammenhang mit „Pipelines“ unter „Latency“ und „Throughput“? Beschreiben Sie die beiden Begriffe an Hand eines einfachen Beispiels. (10 Punkte)

4. Gegeben sei eine PDP-8-Maschine mit einem Cachespeicher, welcher für 4 Speicherworte Platz hat. Es handelt sich dabei um einen Cachespeicher mit „Direct Mapping“. Der anfängliche Inhalt des Cache-Speichers ist rechts abgebildet (in Oktalcode notiert). Im Hauptspeicher steht ein Programm, welches ebenfalls rechts in Oktalcode abgebildet ist. Zeigen Sie die Änderung des Cache-Speichers bei Exekution des Programmes im Hauptspeicher und erläutern Sie die Funktion des Cache-Speichers. Es reicht die Untersuchung der ersten 5 Instruktionen. (10 Punkte)

0000/700
0001/1006
0002/1011
0003/7510
0004/5002
0005/7402
0006/7760
0011/0002

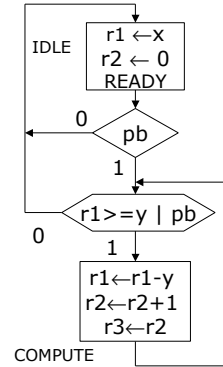
Mnemonic	Code	Semantics
AND	0xxxg	ac ← ac & m[xxxg]
TAD	1xxxg	{link,ac} ← {link,ac} + m[xxxg];
ISZ	2xxxg	m[xxxg] ← m[xxxg]+1; if (m[xxxg]==0) skip next instruction
DCA	3xxxg	m[xxxg] ← ac; ac ← 0;
JMS	4xxxg	Jump to subroutine at xxxg; save return address at xxxg; start execution at xxxg+1
JMP	5xxxg	Jump to memory address xxxg;
CLA	7200g	ac ← 0
CLL	7100g	link ← 0
CMA	7040g	ac ← ~ac
CML	7020g	link ← ~link
RAR	7010g	rotate right {link, ac}
RAL	7004g	rotate left {link, ac}
IAC	7001g	{link, ac} ← ac + 1
SMA	7500g	Skip if accumulator is minus
SPA	7510g	Skip if accumulator is positive
SZA	7440g	Skip if accumulator is zero
SNA	7450g	Skip if accumulator is non-zero
SZL	7430g	Skip if link is zero
SNL	7420g	Skip if link is non-zero
HLT	7402g	Halt the computer
OSR	7404g	ac ← ac sr

tag	data
0004	5002
0011	0002
0006	7760
0003	7510

Schlüssel: 0-20 Punkte: Nicht Genügend; 21-25 Punkte: Genügend; 26-30 Punkte: Befriedigend; 31-35 Punkte: Gut; 36-40 Punkte: Sehr Gut.

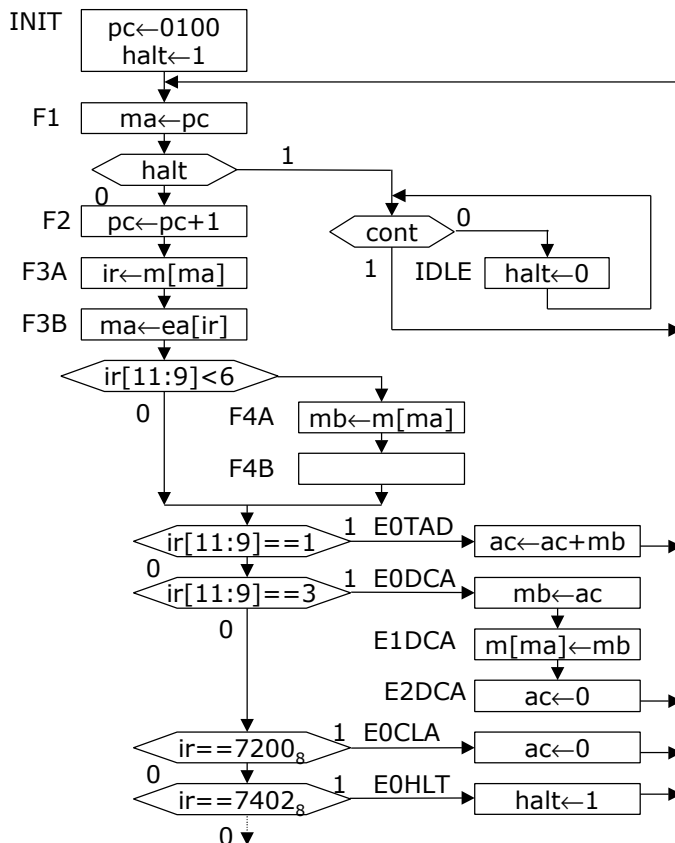
1. Wandeln Sie die Zahl $17/7$ in Binärdarstellung um. Zeigen Sie zumindest drei Stellen rechts vom Komma. Ist die Zahl periodisch? Erläutern Sie die Umrechnung (10 Punkte)

2. Entwerfen Sie einen Datenpfad für die rechts abgebildete ASM. Die zur Verfügung stehenden Module sind Register, Addierer, Subtrahierer und Zähler. Beschreiben Sie Eingänge und Ausgänge des Daten-Interfaces und des Kontroll-Interfaces Ihrer Schaltung. Erläutern Sie Ihre Lösung. (10 Punkte)



3. Was versteht man unter Setup-Zeit und Hold-Zeit bei einem Flipflop? Was ist die Delay-Zeit? Wie groß darf die kombinatorische Verzögerung eines Automaten sein, wenn zusätzlich zu diesen drei Zeiten die Taktperiode gegeben ist? (10 Punkte)

4. Entwerfen Sie die Schaltung rund um das Register ma (in Registertransferbeschreibung) und zeigen Sie ein Strukturbild dieser Schaltung. Skizzieren Sie ein Verilog-Modell Ihrer Schaltung. Die rechtsstehende ASM dient dazu als Grundlage. Woher kommen die Signale der Eingänge Ihrer Schaltung? Erläutern Sie Ihre Lösung. (10 Punkte)

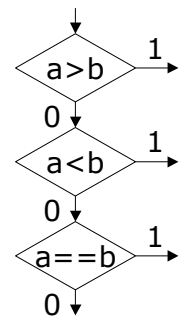


Viel Erfolg!

Schlüssel: 0-20 Punkte: Nicht Genügend; 21-25 Punkte: Genügend; 26-30 Punkte: Befriedigend;
31-35 Punkte: Gut; 36-40 Punkte: Sehr Gut.

1. Gegeben sind zwei 6-Bit-Register A und B mit den Inhalten $A = (07)_8$, $B = (60)_8$ (oktal notiert). Zeigen Sie eine Schaltung, mit deren Hilfe man die Summe $A+B$ bildet und in C speichert. Zur Verfügung stehen Volladdierer und Flipflops. Interpretieren Sie den Inhalt von C unter der Annahme, dass es sich um vorzeichenlose (unsigned) Binärzahlen handelt. Wie sieht das aus, wenn es sich um vorzeichenbehaftete (signed) Zahlen handelt? Erläutern Sie Ihre Antwort. (10 Punkte)

2. Rechts sehen Sie einen Ausschnitt aus einer ASM. Die Werte von a und b sind Registerinhalte. Welche Schaltung würde dieser Teil der ASM in einem Datenpfad implizieren? Modellieren Sie diesen Teil des Datenpfades als „module“ in Verilog. (10 Punkte)



3. Überlegen Sie eine dreistufige Pipeline-Implementierung einer PDP-8. Die drei Pipeline-Stufen sind „Fetch Instruction“, „Fetch Operand“ und „Execute Instruction“. Erläutern Sie die dabei typisch auftretenden Probleme. (10 Punkte)

4. Übersetzen Sie das folgende Programm in PDP-8-Assemblercode:
`a = 10;`
`b = 1;`
`while (a < b) {`
`a = a - 1;`
`}`

Erläutern Sie Ihre Lösung. (10 Punkte)

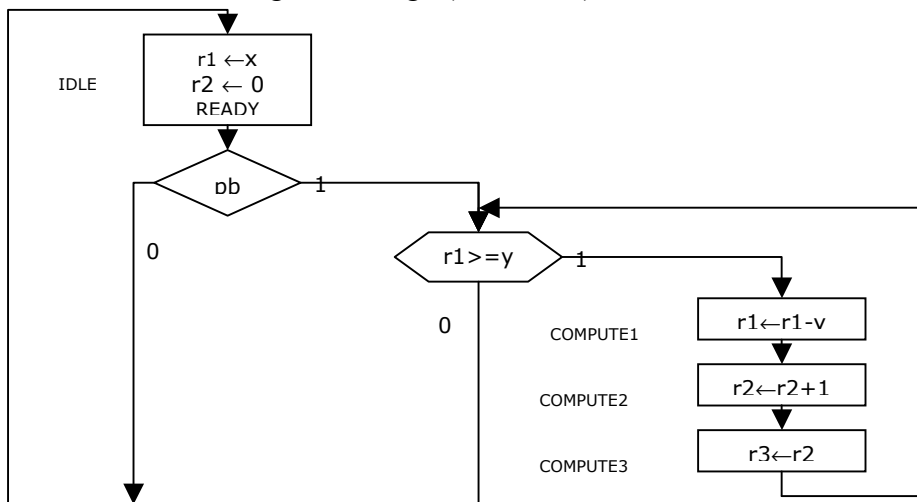
Mnemonic	Code	Semantics
AND	0xxxg	$ac \leftarrow ac \& m[xxxg]$
TAD	1xxxg	$\{link, ac\} \leftarrow \{link, ac\} + m[xxxg];$
ISZ	2xxxg	$m[xxxg] \leftarrow m[xxxg] + 1; \text{ if } (m[xxxg] == 0) \text{ skip next instruction}$
DCA	3xxxg	$m[xxxg] \leftarrow ac; ac \leftarrow 0;$
JMS	4xxxg	Jump to subroutine at xxxg; save return address at xxxg; start execution at xxxg+1
JMP	5xxxg	Jump to memory address xxxg;
CLA	7200g	$ac \leftarrow 0$
CLL	7100g	$link \leftarrow 0$
CMA	7040g	$ac \leftarrow \sim ac$
CML	7020g	$link \leftarrow \sim link$
RAR	7010g	rotate right {link, ac}
RAL	7004g	rotate left {link, ac}
IAC	7001g	$\{link, ac\} \leftarrow ac + 1$
SMA	7500g	Skip if accumulator is minus
SPA	7510g	Skip if accumulator is positive
SZA	7440g	Skip if accumulator is zero
SNA	7450g	Skip if accumulator is non-zero
SZL	7430g	Skip if link is zero
SNL	7420g	Skip if link is non-zero
HLT	7402g	Halt the computer
OSR	7404g	$ac \leftarrow ac sr$

Viel Erfolg!

Schlüssel: 0-20 Punkte: Nicht Genügend; 21-25 Punkte: Genügend; 26-30 Punkte: Befriedigend;
31-35 Punkte: Gut; 36-40 Punkte: Sehr Gut.

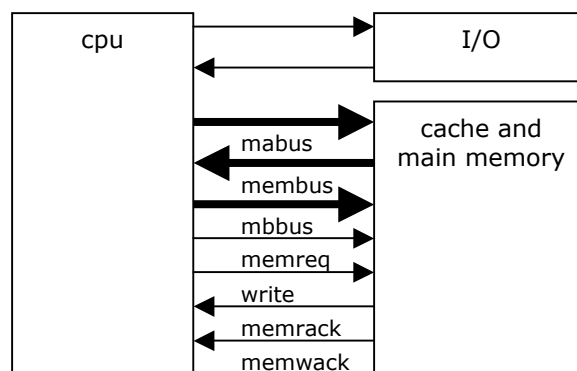
1. Gegeben sind zwei 4-Bit-Register A und B mit den Inhalten $A = 0x3$, $B = 0x9$ (hexadezimal notiert). Zeigen Sie eine Schaltung, mit deren Hilfe man die Summe $A+B$ bildet und in C speichert. Zur Verfügung stehen Volladdierer und Flipflops. Interpretieren Sie den Inhalt von C unter der Annahme, dass es sich um vorzeichenlose (unsigned) Binärzahlen handelt. Wie sieht das aus, wenn es sich um vorzeichenbehaftete (signed) Zahlen handelt? Erläutern Sie Ihre Antwort. (10 Punkte)

2. Die unten gezeigte ASM soll zur Division x/y verwendet werden. Erklären sie den Algorithmus und warum er eine Division erzeugt. In welchem Fall funktioniert er nicht? Haben sie einen Verbesserungsvorschlag? (10 Punkte)

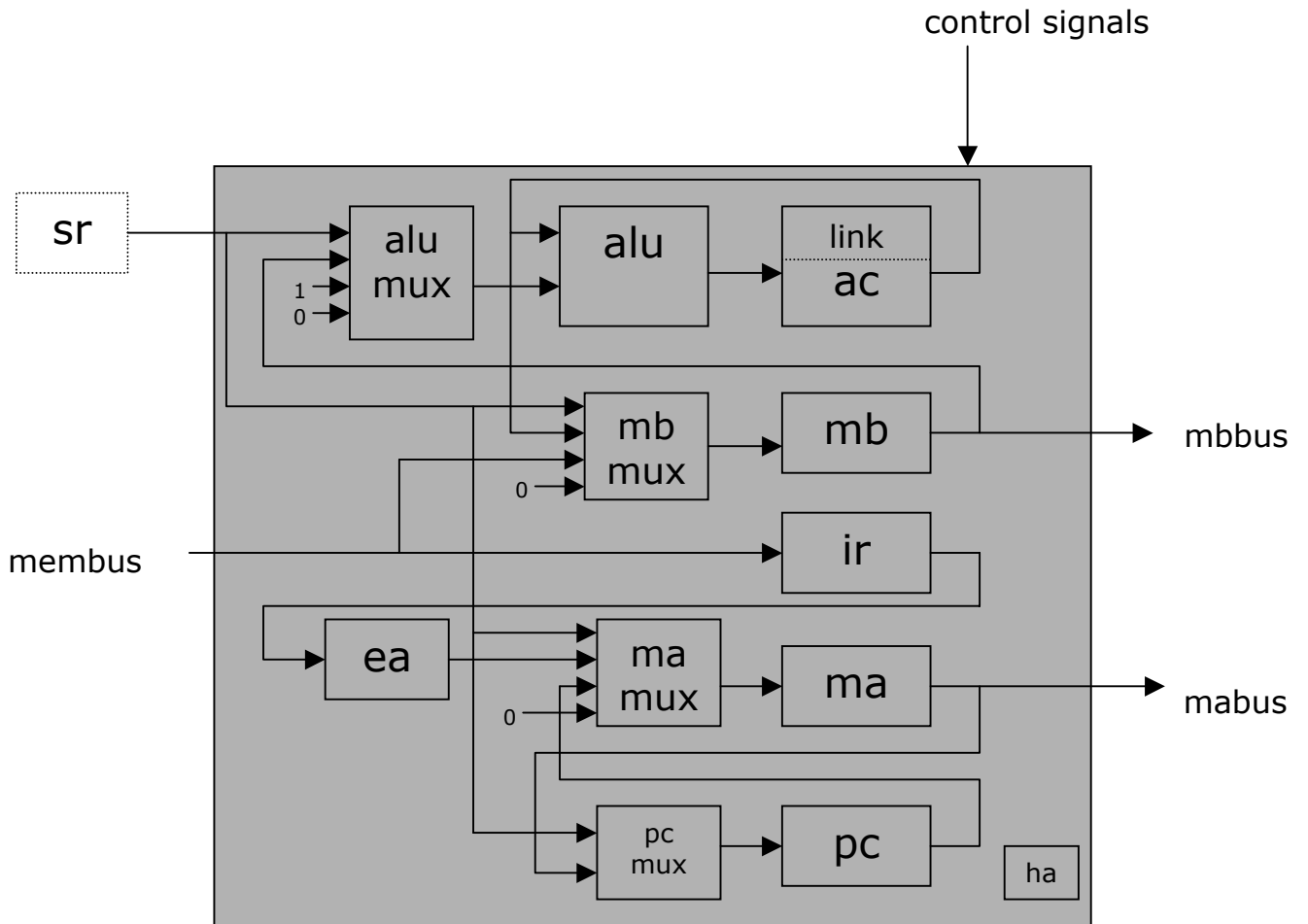
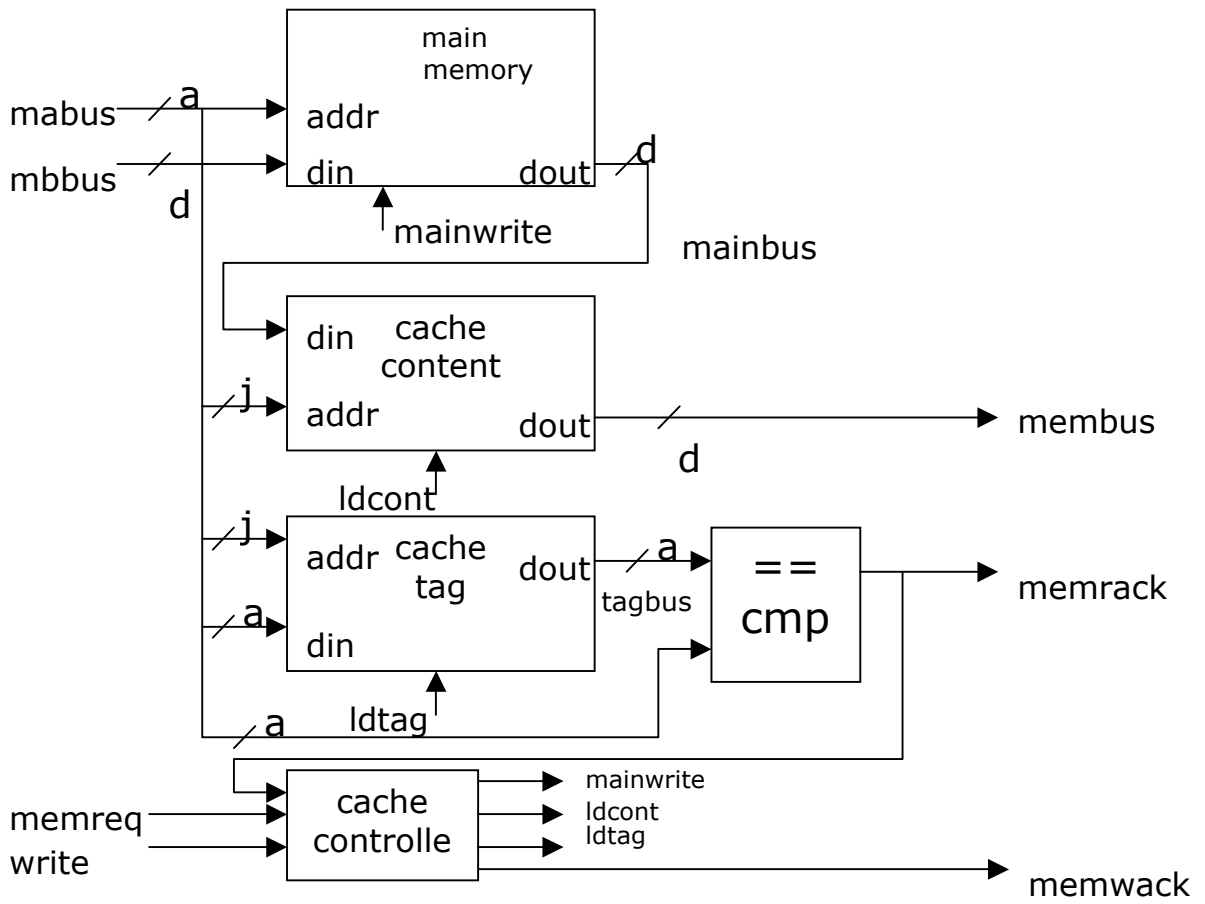


3. Auf Seite 2 finden sie ein Strukturbild des Datenpfads einer PDP8. Beschreiben sie die einzelnen Komponenten des Datenpfades und deren Funktion. Erläutern sie das Zusammenspiel der Komponenten an Hand des Ablaufs einer Instruktion. (10 Punkte)

4. Gegeben sei ein Computersystem mit CPU, I/O, Cache und Hauptspeicher (main memory). Das Blockschaltbild ist unten gezeigt. Auf Seite 2 finden sie ein Strukturbild des Blockes „cache and main memory“. Erläutern sie, wie der die CPU Instruktionen bzw. Daten aus dem Speicher holt und wie dabei der Cache mitspielt anhand der beiden Bilder. (10 Punkte)



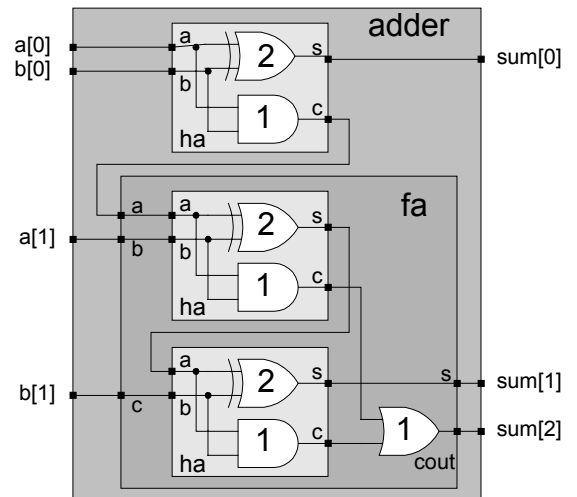
Viel Erfolg!



Schlüssel: 0-20 Punkte: Nicht Genügend; 21-25 Punkte: Genügend; 26-30 Punkte: Befriedigend;
31-35 Punkte: Gut; 36-40 Punkte: Sehr Gut.

1. Erklären sie das Prinzip des r-Komplements. Wozu dient es? Warum funktioniert es? Geben sie Beispiele mit $r=10$ und $r=2$. (10 Punkte)

2. Diskutieren sie an Hand der nebenstehenden Schaltung das Phänomen der kombinatorischen Verzögerung (Delay). Worin besteht hinsichtlich der Maximalgeschwindigkeit die Einschränkung bei dieser Schaltung? (10 Punkte)



3. Was versteht man im Zusammenhang mit „Pipelines“ unter „Latency“ und „Throughput“? Beschreiben Sie die beiden Begriffe an Hand eines einfachen Beispiels. (10 Punkte)

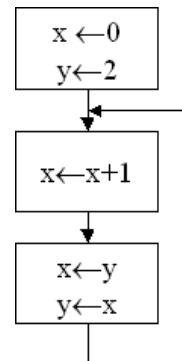
4. Entwerfen sie eine „PDP-8 Light“; dieser Spaßcomputer hat nur zwei Instruktionen. Zeigen sie zuerst die ASM ihres Vorschlages. Skizzieren sie danach den Datenpfad („Architektur“). Erläutern sie die Funktionsweise ihrer Lösung. (10 Punkte)

Mnemonic	Code	Semantics
TAD	1xxx8	{link,ac} ← {link,ac} + m[xxx8];
DCA	3xxx8	m[xxx8] ← ac; ac ← 0;

Viel Erfolg!

Schlüssel: 0-20 Punkte: Nicht Genügend; 21-25 Punkte: Genügend; 26-30 Punkte: Befriedigend;
31-35 Punkte: Gut; 36-40 Punkte: Sehr Gut.

1. Entwerfen sie das Zeitdiagramm für die neben dargestelltenn ASMs. x, y und z sind 8-Bit-Register. (10 Punkte)



2. Zeigen sie das Blockdiagramm eines Datenpfades (architecture) zur rechts dargestellten ASM. (10 Punkte)

3. Entwerfen Sie ein funktionales Verilog-Modell für einen 8-Bit-Auf/Ab-Zähler. Das Interface des Moduls sollte folgendermaßen aussehen:

```
module updown_register(din, dout, ld, up, count, clk);
```

din ... Paralleler Eingang

dout ... Ausgang

ld ... wenn gleich 1, dann laden

up ... wenn gleich 1, dann aufwärts zählen

count ... wenn gleich 1, dann zählen

clk ... Takteingang

Gibt es Mehrdeutigkeiten in der Angabe des Funktion des Interfaces? Denken sie darüber nach und geben sie dazu eine Antwort. Haben sie an den Anfangszustand gedacht? (10 Punkte)

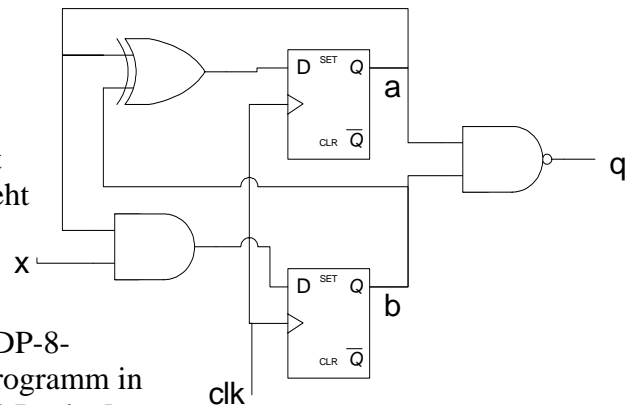
4. Skizzieren sie für den Auf/Ab-Zähler aus Beispiel 3 eine Struktur auf Registertransferebene. (Register, Multiplexer, Addierer, etc.) (10 Punkte)

Viel Erfolg!

Schlüssel: 0-20 Punkte: Nicht Genügend; 21-25 Punkte: Genügend;
26-30 Punkte: Befriedigend; 31-35 Punkte: Gut; 36-40 Punkte: Sehr Gut.

1. Erklären Sie den Unterschied zwischen den zwei Kommunikationsmöglichkeiten „Enable-Handshake“ und „Full-Handshake“. Wie funktionieren beide Varianten? [10 Punkte]
2. Zeigen Sie die Wahrheitstabelle eines 2-zu-1-Multiplexers. Implementieren Sie die Wahrheitstafel des 2-zu-1-Multiplexers mit Hilfe von Mintermen und einer Oder-Verknüpfung. [10 Punkte]

3. Gegeben ist die rechts gezeigte Schaltung. Ermitteln Sie die Zustandsübergangstabelle dieser Schaltung. Ermitteln Sie die Ausgangstabelle dieser Schaltung. Wie sieht die Zustandsübergangsfunktion aus? Wie sieht die Ausgangslogikfunktion aus? Erläutern Sie Ihre Lösung. [10 Punkte]



4. Übersetzen sie das folgende Programm in PDP-8-Assemblercode. Zeigen sie das Maschinenprogramm in Oktalnotation. Erläutern sie ihre Lösung. [10 Punkte]

```

A = 15;
B = 3;
C = 1;
do {
    A = A - B;
    C = 2*C;
} while (A >= C);
    
```

Mnemonic	Code	Semantics
AND	0xxxg	Ac = ac & m[xxxg]
TAD	1xxxg	{link,ac} = {link,ac} + m[xxxg];
ISZ	2xxxg	m[xxxg] = m[xxxg]+1; if (m[xxxg]==0) skip next instruction
DCA	3xxxg	m[xxxg] = ac; ac = 0;
JMS	4xxxg	Jump to subroutine at xxxg; save return address at xxxg; start execution at xxxg+1
JMP	5xxxg	Jump to memory address xxxg;
CLA	7200g	ac = 0
CLL	7100g	link = 0
CMA	7040g	ac = ~ac
CML	7020g	link = ~link
RAR	7010g	Rotate right {link, ac}
RAL	7004g	Rotate left {link, ac}
IAC	7001g	{link, ac} = ac + 1
SMA	7500g	Skip if accumulator is minus
SPA	7510g	Skip if accumulator is positive
SZA	7440g	Skip if accumulator is zero
SNA	7450g	Skip if accumulator is non-zero
SZL	7430g	Skip if link is zero
SNL	7420g	Skip if link is non-zero
HLT	7402g	Halt the computer
OSR	7404g	ac ~ ac sr

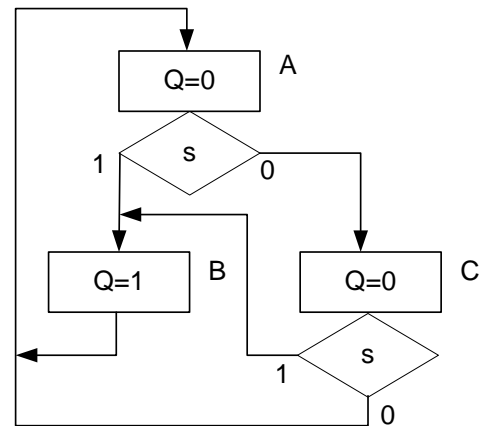
Viel Erfolg!

Schlüssel: 0-20 Punkte: Nicht Genügend; 21-25 Punkte: Genügend;
26-30 Punkte: Befriedigend; 31-35 Punkte: Gut; 36-40 Punkte: Sehr Gut.

1. Was versteht man unter „Denken in Schichten“ (Thinking in Layers)? Welche Schichten haben wir in der Lehrveranstaltung definiert? Beschreiben Sie die Merkmale der Schichten. Warum sind Schichten sinnvoll? Worin besteht der Zusammenhang der Schichten? Wie kommt man von einer Schicht zur nächsten? [10 Punkte]

2. Beschreiben Sie das logische Verhalten eines Dekoders. In welchem Zusammenhang steht er mit dem Begriff „Minterm“? Zeigen Sie einen möglichen inneren Aufbau eines Dekoders mit 4 Ausgängen. [10 Punkte]

3. Ermitteln Sie die Logikgleichungen für die Next-State-Logic und die Output-Logic für die rechts dargestellte ASM. Zeichnen sie ein Diagramm bestehend aus Flipflop(s) und Logikfunktionen. Der Anfangszustand sei der Zustand A. Dieser soll mit Hilfe eines asynchronen Rücksetzeingangs gesetzt werden können. [10 Punkte]



4. Übersetzen sie das folgende Programm in PDP-8-Assemblercode. Zeigen sie das Maschinenprogramm in Oktalnotation. Erläutern sie ihre Lösung. [10 Punkte]

```

A = 15;
B = 3;
C = 1;
while (A != B) {
    B = A + B;
    C = 2*C ;
}
    
```

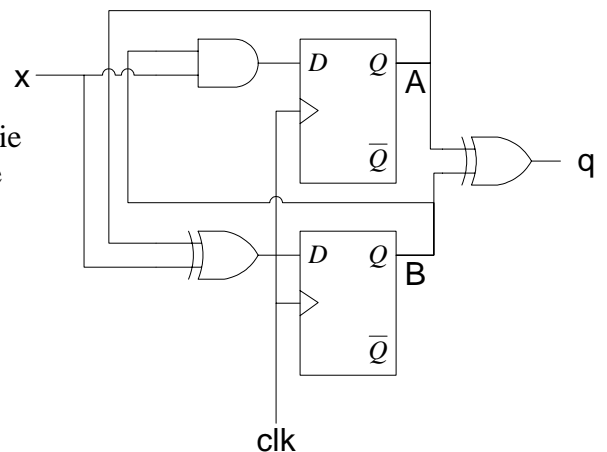
Viel Erfolg!

Mnemonic	Code	Semantics
AND	0xxxg	ac = ac & m[xxxg]
TAD	1xxxg	{link,ac} = {link,ac} + m[xxxg];
ISZ	2xxxg	m[xxxg] = m[xxxg]+1; if (m[xxxg]==0) skip next instruction
DCA	3xxxg	m[xxxg] = ac; ac = 0;
JMS	4xxxg	Jump to subroutine at xxxg; save return address at xxxg; start execution at xxxg+1
JMP	5xxxg	Jump to memory address xxxg;
CLA	7200g	ac = 0
CLL	7100g	link = 0
CMA	7040g	ac = ~ac
CML	7020g	link = ~link
RAR	7010g	Rotate right {link, ac}
RAL	7004g	Rotate left {link, ac}
IAC	7001g	{link, ac} = ac + 1
SMA	7500g	Skip if accumulator is minus
SPA	7510g	Skip if accumulator is positive
SZA	7440g	Skip if accumulator is zero
SNA	7450g	Skip if accumulator is non-zero
SZL	7430g	Skip if link is zero
SNL	7420g	Skip if link is non-zero
HLT	7402g	Halt the computer
OSR	7404g	ac ~ ac sr

Schlüssel: 0-20 Punkte: Nicht Genügend; 21-25 Punkte: Genügend;
 26-30 Punkte: Befriedigend; 31-35 Punkte: Gut; 36-40 Punkte: Sehr Gut.

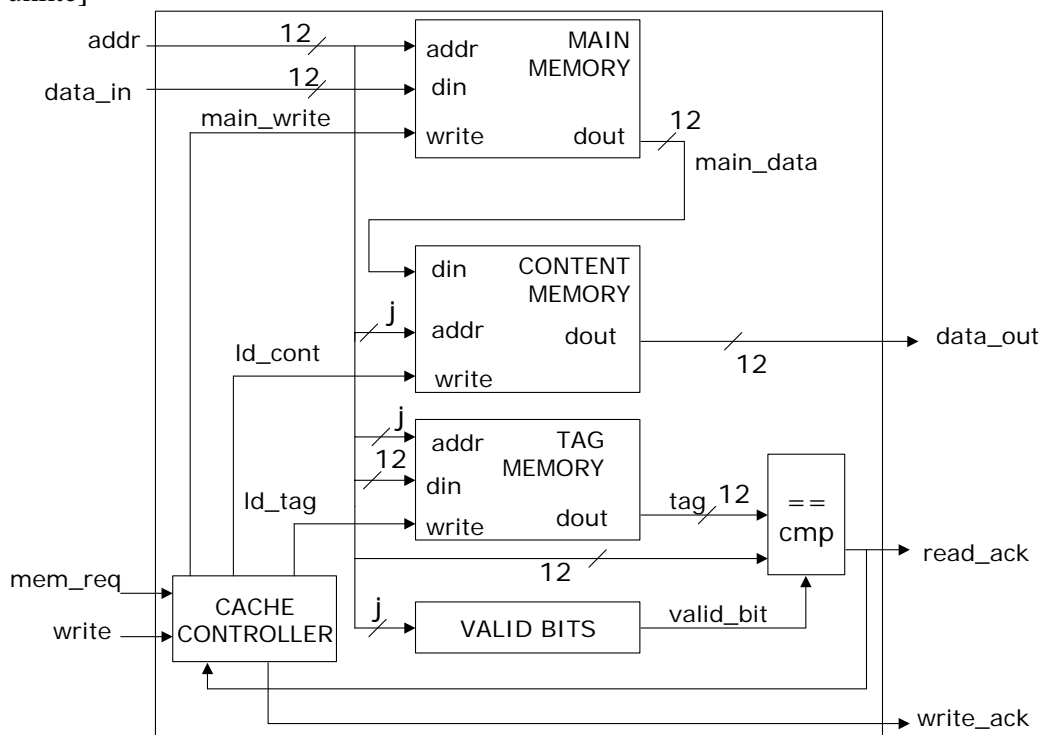
1. Was versteht man unter „Implicit State Encoding“? Geben Sie ein Beispiel dafür (unter Verwendung von Verilog). Geben Sie auch ein Beispiel für eine Modellierung, welche eine Schicht tiefer liegt (ebenfalls unter Verwendung von Verilog). Vergleichen Sie die beiden Modellierungsvarianten. [10 Punkte]
2. Erläutern Sie das Konzept „Interrupt“. Wie ist es bei der PDP-8 implementiert? Was ist ein Interrupt-Service-Programm? Welche Unterschiede und Gemeinsamkeiten gibt es zu einem Unterprogrammaufruf? Geben Sie Beispiele für den Einsatz von Interrupts. Sehen Sie einen Zusammenhang zwischen Interrupts und der Kommunikation zwischen Modulen? [10 Punkte]
3. Erläutern Sie das Konzept „Master-Slave-Flipflop“. Zeigen Sie eine Schaltung unter Verwendung von zwei Multiplexern und einem Inverter? Wozu braucht man Master-Slave-Flipflops? [10 Punkte]

4. Gegeben ist die nebenan gezeigte Schaltung. Ermitteln Sie die Zustandsübergangstabelle dieser Schaltung. Ermitteln Sie die Ausgangstabelle dieser Schaltung. Wie sieht die Zustandsübergangsfunktion aus? Wie sieht die Ausgangslogikfunktion aus? Erläutern Sie Ihre Lösung. [10 Punkte]



Schlüssel: 0-20 Punkte: Nicht Genügend; 21-25 Punkte: Genügend;
 26-30 Punkte: Befriedigend; 31-35 Punkte: Gut; 36-40 Punkte: Sehr Gut.

1. Erklären Sie die Begriffe Eingang, Ausgang, Zustand und Zustandsübergang bei einem endlichen Automaten. Setzen Sie diese Begriffe in Beziehung zueinander. Welche Möglichkeiten kennen Sie, endliche Automaten zu beschreiben (Welche haben wir in der Lehrveranstaltung verwendet)? Verwenden Sie ein Beispiel zur Erläuterung Ihrer Antworten [10 Punkte]
2. Sie haben ein Register mit 12 Bit Breite und möchten darin eine Festkommazahl in Zweierkomplementdarstellung speichern. Verwenden sie 8 Bit rechts vom Komma („Binärpunkt“). Welches ist die größte positive Zahl, welche in diesem Format darstellbar ist? Welches ist die größte negative Zahl? Stellen sie die Zahl $13/7$ in diesem Register dar. Erläutern sie Ihre Antwort. Stellen Sie $-13/7$ dar. [10 Punkte]
3. Erklären Sie die Funktionsweise der unten gezeigten Schaltung. Wozu dient sie? [10 Punkte]



4. Zeigen sie den Aufbau eines synchronen RAMs mit der Größe 2×2 Bit. Verwenden sie Flipflops, Multiplexer, Gatter, etc. Erklären sie die Funktionsweise ihrer Schaltung. [10 Punkte]

Viel Erfolg!

Schlüssel: 0-20 Punkte: Nicht Genügend; 21-25 Punkte: Genügend;
26-30 Punkte: Befriedigend; 31-35 Punkte: Gut; 36-40 Punkte: Sehr Gut.

1. Erklären Sie die Begriffe Setup-Zeit, Hold-Zeit und Verzögerungszeit bei Flipflops. Zeigen Sie deren Bedeutung in Zusammenhang mit der Realisierung von endlichen Automaten. Wie hängt dies alles mit der maximalen Taktfrequenz einer Schaltung zusammen? [10 Punkte]
2. Entwerfen Sie den Datenpfad für eine Schaltung, welche einen 4-Bit-Dateneingang und einen 4-Bit-Datenausgang hat. Setzt man das Eingangssignal „pb“ (push button) auf 1, so startet die Maschine mit dem Laden von 4 am Eingang anliegenden 4-Bit-Werten. Nach Laden der Werte wird der Ausgang „ready“ auf 1 gesetzt und die Werte erscheinen in umgekehrter Reihenfolge am 4-Bit-Ausgang. Verwenden sie Register, Multiplexer, Addierer, Subtrahierer, Gatter, etc. Benennen sie die Eingänge und die Ausgänge der Schaltung. Zeigen Sie eine ASM der Kontrollmaschine. [10 Punkte]
3. Welcher wesentliche Sachverhalt wurde mit unten stehender Tabelle in der Lehrveranstaltung erläutert? Beschreiben Sie die darin vorkommenden Begriffe und erklären Sie deren Zusammenhang. [10 Punkte]

cache size	clock cycles	miss	hit	average access cycles	hit ratio
1	1933	170	0	6,00	0,00%
2	1933	170	0	6,00	0,00%
4	1933	170	0	6,00	0,00%
8	1833	150	20	5,41	11,76%
16	1508	85	85	3,50	50,00%
32	1198	23	147	1,68	86,47%
64	1198	23	147	1,68	86,47%
128	1198	23	147	1,68	86,47%

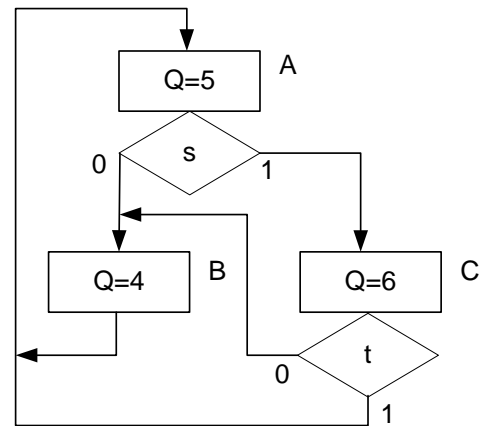
4. Üblicherweise spricht man von Halbaddierern und Volladdierern. Ein Volladdierer zählt etwa 3 Ziffern zusammen. Wie könnte ein Addierer aussehen, welcher 4 Ziffern zusammenzählt? Ergibt ein solcher Addierer einen Sinn? Braucht er einen Übertragseingang (Carry-Input)? Ließe er sich aus anderen Addierern aufbauen? Erläutern Sie auch den Unterschied von Ziffern und Zahlen. Wie könnte man Addierer bauen, welche Zahlen zusammenzählen? [10 Punkte]

Viel Erfolg!

Schlüssel: 0-20 Punkte: Nicht Genügend; 21-25 Punkte: Genügend;
26-30 Punkte: Befriedigend; 31-35 Punkte: Gut; 36-40 Punkte: Sehr Gut.

1. Üblicherweise spricht man von Halbaddierern und Volladdierern. Ein Volladdierer zählt etwa 3 Ziffern zusammen. Wie könnte ein Addierer aussehen, welcher 4 Ziffern zusammenzählt? Ergibt ein solcher Addierer einen Sinn? Ließe er sich aus anderen Addierern aufbauen? [10 Punkte]

2. Was versteht man unter „Implicit State Encoding“?
Geben Sie ein Beispiel dafür (unter Verwendung von Verilog und der Abbildung rechts). Geben Sie auch ein Beispiel für eine Modellierung, welche eine Schicht tiefer liegt (ebenfalls unter Verwendung von Verilog). Vergleichen Sie die beiden Modellierungsvarianten. [10 Punkte]



3. Ermitteln Sie die Logikgleichungen für die Next-State-Logic und die Output-Logic für die rechts oben dargestellte ASM. Zeichnen sie ein Diagramm bestehend aus Flipflop(s) und Logikfunktionen. Der Anfangszustand sei der Zustand A. Dieser soll mit Hilfe eines asynchronen Rücksetzeingangs gesetzt werden können. [10 Punkte]

4. Übersetzen sie das folgende Programm in PDP-8-Assemblercode. Zeigen sie das Maschinenprogramm in Oktalnotation. Erläutern sie ihre Lösung. [10 Punkte]

```

A = 15;
B = 3;
C = 1;
while (A != B) {
    B = A + B;
    C = 2*C ;
}
  
```

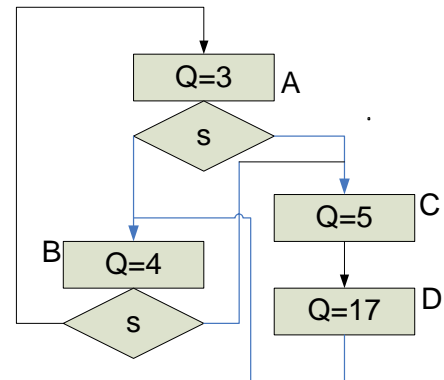
Viel Erfolg!

Mnemonic	Code	Semantics
AND	0xxxg	ac = ac & m[xxxg]
TAD	1xxxg	{link,ac} = {link,ac} + m[xxxg];
ISZ	2xxxg	m[xxxg] = m[xxxg]+1; if (m[xxxg]==0) skip next instruction
DCA	3xxxg	m[xxxg] = ac; ac = 0;
JMS	4xxxg	Jump to subroutine at xxxg; save return address at xxxg; start execution at xxxg+1
JMP	5xxxg	Jump to memory address xxxg;
CLA	7200g	ac = 0
CLL	7100g	link = 0
CMA	7040g	ac = ~ac
CML	7020g	link = ~link
RAR	7010g	Rotate right {link, ac}
RAL	7004g	Rotate left {link, ac}
IAC	7001g	{link, ac} = ac + 1
SMA	7500g	Skip if accumulator is minus
SPA	7510g	Skip if accumulator is positive
SZA	7440g	Skip if accumulator is zero
SNA	7450g	Skip if accumulator is non-zero
SZL	7430g	Skip if link is zero
SNL	7420g	Skip if link is non-zero
HLT	7402g	Halt the computer
OSR	7404g	ac ~ ac sr

Schlüssel: 0-20 Punkte: Nicht Genügend; 21-25 Punkte: Genügend;
26-30 Punkte: Befriedigend; 31-35 Punkte: Gut; 36-40 Punkte: Sehr Gut.

1. Üblicherweise spricht man von Halbaddierern und Volladdierern. Ein Volladdierer zählt etwa 3 Ziffern zusammen. Wie könnte ein Addierer aussehen, welcher 4 Ziffern zusammenzählt? Ließe er sich aus anderen Addierern aufbauen? [10 Punkte]

2. Ermitteln Sie die Logikgleichungen für die Next-State-Logic und die Output-Logic für die rechts dargestellte ASM. Zeichnen sie ein Diagramm bestehend aus Flipflop(s) und Logikgleichungen. Der Anfangszustand sei der Zustand A. Dieser soll mit Hilfe eines asynchronen Rücksetzeingangs gesetzt werden können. [10 Punkte]



3. Diskutieren Sie „Full-Handshake-Kommunikation“. Der Master soll in Ihrem Beispiel als Produzent auftreten. Wie funktioniert diese Kommunikation? Skizzieren Sie ein ASM-Diagramm für den Slave. Welche Vorteile bieten sich gegenüber der Variante, wo lediglich ein Enable-Signal für die Kommunikation verwendet wird? [10 Punkte]

4. Übersetzen sie das folgende Programm in PDP-8-Assemblercode. Zeigen sie das Maschinenprogramm in Oktalnotation. Erläutern sie ihre Lösung. [10 Punkte]

```

A = 15;
B = 3;
C = 1;
while (A != B) {
    B = A + B;
    C = 2*C ;
}
    
```

Viel Erfolg!

Mnemonic	Code	Semantics
AND	0xxxg	ac = ac & m[xxxg]
TAD	1xxxg	{link,ac} = {link,ac} + m[xxxg];
ISZ	2xxxg	m[xxxg] = m[xxxg]+1; if (m[xxxg]==0) skip next instruction
DCA	3xxxg	m[xxxg] = ac; ac = 0;
JMS	4xxxg	Jump to subroutine at xxxg; save return address at xxxg; start execution at xxxg+1
JMP	5xxxg	Jump to memory address xxxg;
CLA	7200g	ac = 0
CLL	7100g	link = 0
CMA	7040g	ac = ~ac
CML	7020g	link = ~link
RAR	7010g	Rotate right {link, ac}
RAL	7004g	Rotate left {link, ac}
IAC	7001g	{link, ac} = ac + 1
SMA	7500g	Skip if accumulator is minus
SPA	7510g	Skip if accumulator is positive
SZA	7440g	Skip if accumulator is zero
SNA	7450g	Skip if accumulator is non-zero
SZL	7430g	Skip if link is zero
SNL	7420g	Skip if link is non-zero
HLT	7402g	Halt the computer
OSR	7404g	ac ~ ac sr

Schlüssel: 0-20 Punkte: Nicht Genügend; 21-25 Punkte: Genügend;
26-30 Punkte: Befriedigend; 31-35 Punkte: Gut; 36-40 Punkte: Sehr Gut.

1. Beschreiben Sie das logische Verhalten eines Dekoders. In welchem Zusammenhang steht er mit dem Begriff „Minterm“? Zeigen sie einen möglichen inneren Aufbau eines Dekoders mit 4 Ausgängen. [10 Punkte]
2. Entwerfen ein ASM-Diagramm einer 1-Byte-FIFO. Die FIFO habe einen 8-Bit-Dateneingang DIN und einen 8-Bit-Datenausgang DOUT. Intern kann die FIFO genau 1 Byte speichern. Der Datenproduzent und der Datenkonsument werden synchron mit der FIFO getaktet. Die FIFO besitzt einen Ausgang FULL, welcher signalisiert, dass ein Byte vom Produzenten in die FIFO geschrieben wurde, jedoch vom Konsumenten noch nicht abgeholt wurde. Mit dem Signal PUT kann der Produzent ein Datum in die FIFO schreiben. Mit dem SIGNAL GET kann der Konsument ein Datum aus der FIFO lesen. Es obliegt der Verantwortung des Produzenten und des Konsumenten, das Signal FULL sinnvoll zu interpretieren. Erläutern Sie Ihre Lösung. [10 Punkte]
3. Gegeben sei ein 3-Bit-Addierer. Dieser setzt sich aus 3 Volladdierern, welche in Kette geschaltet sind, zusammen. Jeder Volladdierer habe eine Verzögerungszeit von 1 Zeiteinheit von Eingang zu Ausgang. Welche Verzögerungszeit entsteht für die Addition von 3+4, wenn zuvor 0+0 addiert wurde? Was ist die maximale Verzögerungszeit des 3-Bit-Addierers? Welche Situation von Eingangsdaten lässt diese maximale Verzögerungszeit entstehen? [10 Punkte]
4. Erläutern Sie das Konzept „Interrupt“. Wie ist es bei der PDP-8 implementiert? Was ist ein Interrupt-Service-Programm? Welche Unterschiede und Gemeinsamkeiten gibt es zu einem Unterprogrammaufruf? Geben Sie Beispiele für den Einsatz von Interrupts. Sehen Sie einen Zusammenhang zwischen Interrupts und der Kommunikation zwischen Modulen? [10 Punkte]

Viel Erfolg!

Schlüssel: 0-20 Punkte: Nicht Genügend; 21-25 Punkte: Genügend;
26-30 Punkte: Befriedigend; 31-35 Punkte: Gut; 36-40 Punkte: Sehr Gut.

1. Erläutern Sie das Konzept „Master-Slave-Flipflop“. Zeigen Sie eine Schaltung unter Verwendung von Multiplexern und anderen Gattern. Wozu braucht man Master-Slave-Flipflops? [10 Punkte]
2. Entwerfen ein ASM-Diagramm einer 1-Byte-FIFO. Die FIFO habe einen 4-Bit-Dateneingang DIN und einen 8-Bit-Datenausgang DOUT. Intern kann die FIFO genau 1 Byte speichern. Der Datenproduzent und der Datenkonsument werden synchron mit der FIFO getaktet. Der Datenproduzent braucht jeweils zwei hintereinander folgende Taktzyklen, in denen er ein Byte (= 8 Bit) in zwei Hälften (zuerst die niederwertige, dann die hochwertige) zur Verfügung stellt. Die FIFO besitzt einen Ausgang FULL, welcher signalisiert, dass ein Byte vom Produzenten in die FIFO geschrieben wurde, jedoch vom Konsumenten noch nicht abgeholt wurde. Mit dem Signal PUT kann der Produzent ein Datum in die FIFO schreiben. Mit dem SIGNAL GET kann der Konsument ein Datum aus der FIFO lesen. Es obliegt der Verantwortung des Produzenten und des Konsumenten, das Signal FULL sinnvoll zu interpretieren. Erläutern Sie Ihre Lösung. [10 Punkte]
3. Wie sieht die Grundzelle eines Multiplizierers aus? Bauen Sie daraus einen 2-mal-2-Bit-Multiplizierer. Erläutern Sie Ihre Lösung. [10 Punkte]
4. Übersetzen sie das folgende Programm in PDP-8-Assemblercode. Zeigen sie das Maschinenprogramm in Oktalnotation. Erläutern sie ihre Lösung. [10 Punkte]

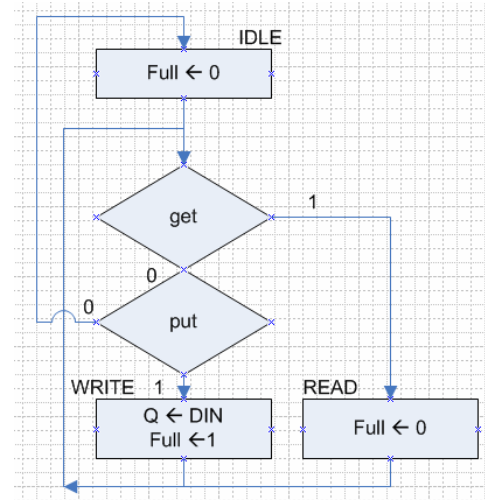
```
A = 15;
B = 1;
C = 3;
do {
    B = A + B;
    C = 3*C ;
} while (A > 2*B);
```

Viel Erfolg!

Mnemonic	Code	Semantics
AND	0xxxg	ac = ac & m[xxxg]
TAD	1xxxg	{link,ac} = {link,ac} + m[xxxg];
ISZ	2xxxg	m[xxxg] = m[xxxg]+1; if (m[xxxg]==0) skip next instruction
DCA	3xxxg	m[xxxg] = ac; ac = 0;
JMS	4xxxg	Jump to subroutine at xxxg; save return address at xxxg; start execution at xxxg+1
JMP	5xxxg	Jump to memory address xxxg;
CLA	7200g	ac = 0
CLL	7100g	link = 0
CMA	7040g	ac = ~ac
CML	7020g	link = ~link
RAR	7010g	Rotate right {link, ac}
RAL	7004g	Rotate left {link, ac}
IAC	7001g	{link, ac} = ac + 1
SMA	7500g	Skip if accumulator is minus
SPA	7510g	Skip if accumulator is positive
SZA	7440g	Skip if accumulator is zero
SNA	7450g	Skip if accumulator is non-zero
SZL	7430g	Skip if link is zero
SNL	7420g	Skip if link is non-zero
HLT	7402g	Halt the computer
OSR	7404g	ac ~ ac sr

Schlüssel: 0-20 Punkte: Nicht Genügend; 21-25 Punkte: Genügend;
26-30 Punkte: Befriedigend; 31-35 Punkte: Gut; 36-40 Punkte: Sehr Gut.

1. Entwerfen Sie einen Datenpfad für das rechts stehende ASM-Diagramm. Es handelt sich um eine einfache synchrone FIFO. Wie sieht das ASM-Diagramm der dazu passenden Kontroll-Logik aus? Erläutern sie die Funktionsweise dieser FIFO. [10 Punkte]
2. Was versteht man unter „Implicit State Encoding“? Zeigen Sie ein Verilog-Modell, welches das rechts gezeigte ASM-Diagramm mit dieser Methode modelliert. Erläutern Sie Ihre Lösung mit Hilfe eines Zeitdiagramms. [10 Punkte]



3. Was versteht man unter „Latenzzeit“, „Durchsatz“? Erläutern Sie die beiden Konzepte an Hand eines Pipeline-Multiplizierers (4*4-Bit). [10 Punkte]
4. Übersetzen Sie die unten gezeigte For-Schleife in ein PDP-8-Assemblerprogramm. Übersetzen sie das Programm in Maschinencode beginnend ab Adresse 0. [10 Punkte]

for (i = 0; i<16; i++) ;

Mnemonic	Code	Semantics
AND	0xxxg	ac = ac & m[xxxg]
TAD	1xxxg	{link,ac} = {link,ac} + m[xxxg];
ISZ	2xxxg	m[xxxg] = m[xxxg]+1; if (m[xxxg]==0) skip next instruction
DCA	3xxxg	m[xxxg] = ac; ac = 0;
JMS	4xxxg	Jump to subroutine at xxxg; save return address at xxxg; start execution at xxxg+1
JMP	5xxxg	Jump to memory address xxxg;
CLA	7200g	ac = 0
CLL	7100g	link = 0
CMA	7040g	ac = ~ac
CML	7020g	link = ~link
RAR	7010g	Rotate right {link, ac}
RAL	7004g	Rotate left {link, ac}
IAC	7001g	{link, ac} = ac + 1
SMA	7500g	Skip if accumulator is minus
SPA	7510g	Skip if accumulator is positive
SZA	7440g	Skip if accumulator is zero
SNA	7450g	Skip if accumulator is non-zero
SZL	7430g	Skip if link is zero
SNL	7420g	Skip if link is non-zero
HLT	7402g	Halt the computer
OSR	7404g	ac = ac sr

Viel Erfolg!

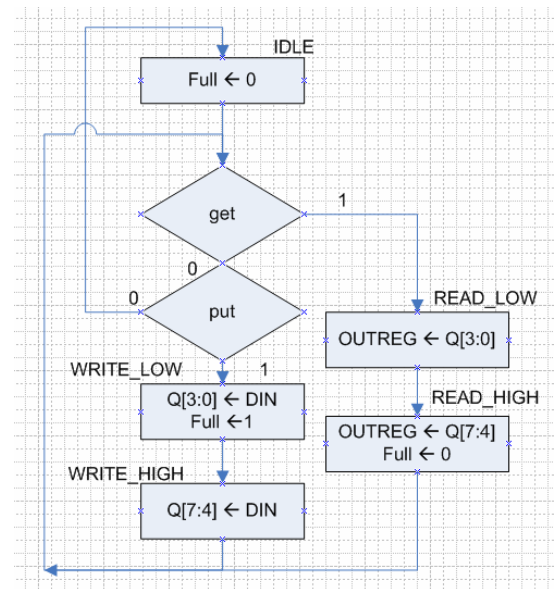
Schlüssel: 0-20 Punkte: Nicht Genügend; 21-25 Punkte: Genügend;
 26-30 Punkte: Befriedigend; 31-35 Punkte: Gut; 36-40 Punkte: Sehr Gut.

1. Entwerfen Sie einen Datenpfad für das rechts stehende ASM-Diagramm. Es handelt sich um eine einfache synchrone FIFO mit 4-Bit-Interfaces zu Produzenten und Konsumenten. Wie sieht das ASM-Diagramm der dazu passenden Kontroll-Logik aus? Erläutern sie die Funktionsweise dieser FIFO. [10 Punkte]

2. Was versteht man unter „Implicit State Encoding“? Zeigen Sie ein Verilog-Modell, welches das rechts gezeigte ASM-Diagramm mit dieser Methode modelliert. Erläutern Sie Ihre Lösung mit Hilfe eines Zeitdiagramms. [10 Punkte]

3. Eine sehr einfache CPU sei mit den 3 Pipelinestufen „Hole Instruktion“, „Hole Datum“ und „Exekutierte Instruktion“ aufgebaut. Was bedeutet das? Warum baut man CPUs mit Pipelines? Welche Probleme entstehen bei der Exekution von Jump-Befehlen und Skip-Befehlen? [10 Punkte]

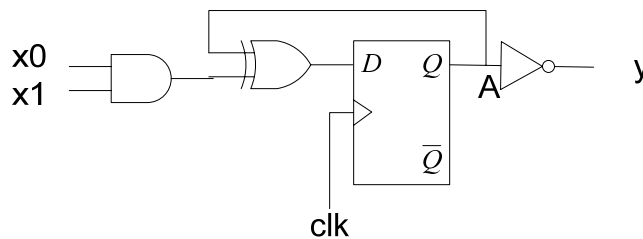
4. Sie haben ein Register mit 10 Bit Breite und möchten darin eine Festkommazahl in Zweierkomplementdarstellung speichern. Verwenden sie 7 Bit rechts vom Komma („Binärpunkt“). Welches ist die größte positive Zahl, welche in diesem Format darstellbar ist? Welches ist die größte negative Zahl? Stellen sie die Zahl $9/7$ in diesem Register dar. Erläutern sie Ihre Antwort. Stellen Sie $-9/7$ dar. [10 Punkte]



Viel Erfolg!

Schlüssel: 0-20 Punkte: Nicht Genügend; 21-25 Punkte: Genügend;
26-30 Punkte: Befriedigend; 31-35 Punkte: Gut; 36-40 Punkte: Sehr Gut.

1. Beschreiben sie das Konzept eines endlichen Automaten (finite state machine). Wählen sie das unten gezeigte Beispiel und zeigen sie an Hand dieses Beispiels die wesentlichen Merkmale von Automaten auf. Welche Hilfsmittel kennen Sie, um diesen Automaten mit Tabellen zu beschreiben? Wie sieht es mit dem Thema „Anfangszustand“ aus? [10 Punkte]



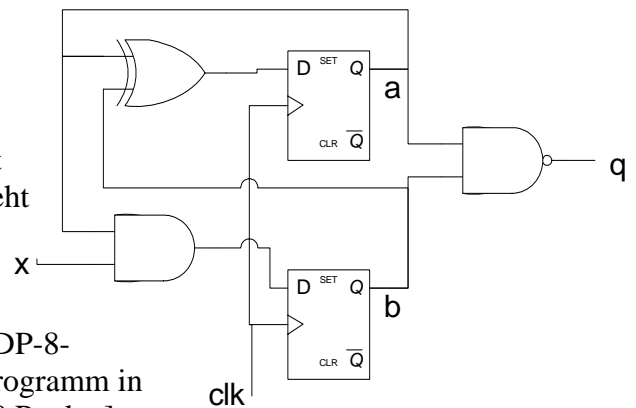
2. Entwerfen ein ASM-Diagramm einer 1-Byte-FIFO. Die FIFO habe einen 8-Bit-Dateneingang DIN und einen 8-Bit-Datenausgang DOUT. Intern kann die FIFO genau 1 Byte speichern. Der Datenproduzent und der Datenkonsument werden synchron mit der FIFO getaktet. Die FIFO besitzt einen Ausgang FULL, welcher signalisiert, dass ein Byte vom Produzenten in die FIFO geschrieben wurde, jedoch vom Konsumenten noch nicht abgeholt wurde. Mit dem Signal PUT kann der Produzent ein Datum in die FIFO schreiben. Mit dem SIGNAL GET kann der Konsument ein Datum aus der FIFO lesen. Es obliegt der Verantwortung des Produzenten und des Konsumenten, das Signal FULL sinnvoll zu interpretieren. Erläutern Sie Ihre Lösung. [10 Punkte]
3. Stellen Sie die Zahl $17/9$ („siebzehn Neuntel“) als Fixkommazahl in Zweierkomplementdarstellung in einem 12-Bit-Register dar. Die 12 Bits sind folgendermaßen aufgeteilt: 3 Bits links vom Komma und 9 Bits rechts vom Komma. Wie wird $-17/9$ dargestellt? Welches sind die größte positive und größte negative darstellbare Zahl in diesem Register (unter Berücksichtigung des „Kommata“). [10 Punkte]
4. Erläutern Sie das Konzept „Interrupt“. Wie ist es bei der PDP-8 implementiert? Was ist ein Interrupt-Servive-Programm? Welche Unterschiede und Gemeinsamkeiten gibt es zu einem Unterprogrammaufruf? Geben Sie Beispiele für den Einsatz von Interrupts. Sehen Sie einen Zusammenhang zwischen Interrupts und der Kommunikation zwischen Modulen? [10 Punkte]

Viel Erfolg!

Schlüssel: 0-20 Punkte: Nicht Genügend; 21-25 Punkte: Genügend;
 26-30 Punkte: Befriedigend; 31-35 Punkte: Gut; 36-40 Punkte: Sehr Gut.

1. Wie sieht die Grundzelle eines Multiplizierers aus? Bauen Sie daraus einen 2-mal-2-Bit-Multiplizierer. Erläutern Sie Ihre Lösung. [10 Punkte]
2. Zeigen Sie die Wahrheitstabelle eines 2-zu-1-Multiplexers. Implementieren Sie die Wahrheitstafel des 2-zu-1-Multiplexers mit Hilfe von Mintermen und einer Oder-Verknüpfung. [10 Punkte]

3. Gegeben ist die rechts gezeigte Schaltung. Ermitteln Sie die Zustandsübergangstabelle dieser Schaltung. Ermitteln Sie die Ausgangstabelle dieser Schaltung. Wie sieht die Zustandsübergangsfunktion aus? Wie sieht die Ausgangslogikfunktion aus? Erläutern Sie Ihre Lösung. [10 Punkte]



4. Übersetzen sie das folgende Programm in PDP-8-Assemblercode. Zeigen sie das Maschinenprogramm in Oktalnotation. Erläutern sie ihre Lösung. [10 Punkte]

```

A = 15;
B = 3;
C = 1;
do {
    A = A - B;
    C = 3*C;
} while (A >= C);
    
```

Mnemonic	Code	Semantics
AND	0xxxg	Ac = ac & m[xxxg]
TAD	1xxxg	{link,ac} = {link,ac} + m[xxxg];
ISZ	2xxxg	m[xxxg] = m[xxxg]+1; if (m[xxxg]=0) skip next instruction
DCA	3xxxg	m[xxxg] = ac; ac = 0;
JMS	4xxxg	Jump to subroutine at xxxg; save return address at xxxg; start execution at xxxg+1
JMP	5xxxg	Jump to memory address xxxg;
CLA	7200g	ac = 0
CLL	7100g	link = 0
CMA	7040g	ac = ~ac
CML	7020g	link = ~link
RAR	7010g	Rotate right {link, ac}
RAL	7004g	Rotate left {link, ac}
IAC	7001g	{link, ac} = ac + 1
SMA	7500g	Skip if accumulator is minus
SPA	7510g	Skip if accumulator is positive
SZA	7440g	Skip if accumulator is zero
SNA	7450g	Skip if accumulator is non-zero
SZL	7430g	Skip if link is zero
SNL	7420g	Skip if link is non-zero
HLT	7402g	Halt the computer
OSR	7404g	ac = ac sr

Viel Erfolg!

Schlüssel: 0-20 Punkte: Nicht Genügend; 21-25 Punkte: Genügend;
26-30 Punkte: Befriedigend; 31-35 Punkte: Gut; 36-40 Punkte: Sehr Gut.

1. Wie sieht die Grundzelle eines Multiplizierers aus? Bauen Sie daraus einen 2-mal-2-Bit-Multiplizierer. Erläutern Sie Ihre Lösung. [10 Punkte]
2. Entwerfen ein ASM-Diagramm einer 1-Byte-FIFO. Die FIFO habe einen 8-Bit-Dateneingang DIN und einen 8-Bit-Datenausgang DOUT. Intern kann die FIFO genau 1 Byte speichern. Der Datenproduzent und der Datenkonsument werden synchron mit der FIFO getaktet. Die FIFO besitzt einen Ausgang FULL, welcher signalisiert, dass ein Byte vom Produzenten in die FIFO geschrieben wurde, jedoch vom Konsumenten noch nicht abgeholt wurde. Mit dem Signal PUT kann der Produzent ein Datum in die FIFO schreiben. Mit dem SIGNAL GET kann der Konsument ein Datum aus der FIFO lesen. Es obliegt der Verantwortung des Produzenten und des Konsumenten, das Signal FULL sinnvoll zu interpretieren. Erläutern Sie Ihre Lösung. [10 Punkte]
3. Was versteht man unter Setup-Zeit, Hold-Zeit und Delay-Zeit? Welche Bedeutung haben diese Zeiten bei der Ermittlung der Maximalgeschwindigkeit des Taktes einer synchronen Schaltung? [10 Punkte]
4. Ermitteln Sie das ASM-Diagramm des unten stehenden Implicit-State-Encoding-Modells. Definieren Sie Eingänge und Ausgänge des Modells. Erläutern Sie, was bei diesem Modell passiert. [10 Punkte]

```
always
begin
    @(posedge clk) enter_new_state(`P_IDLE);
    put <= @(posedge clk) 0;
    i <= @(posedge clk) 0;
    if (start == 1)
        begin
            while (i < 16)
                begin
                    while ($random(j)%3 != 0)
                        @(posedge clk) enter_new_state(`P_NO_SEND);
                    while (ack == 1)
                        @(posedge clk) enter_new_state(`P_WAIT);
                    while (ack == 0)
                        begin
                            @(posedge clk) enter_new_state(`P_SEND1);
                            data <= @(posedge clk) src[i];
                            put <= @(posedge clk) 1;
                        end
                            @(posedge clk) enter_new_state(`P_SEND2);
                            i <= @(posedge clk) i+1;
                            data <= @(posedge clk) 8'bx;
                            put <= @(posedge clk) 0;
                        end
                    while (1)
                        begin
                            @(posedge clk) enter_new_state(`P_READY);
                            ready = 1;
                        end
                end
        end
end
```

Viel Erfolg!