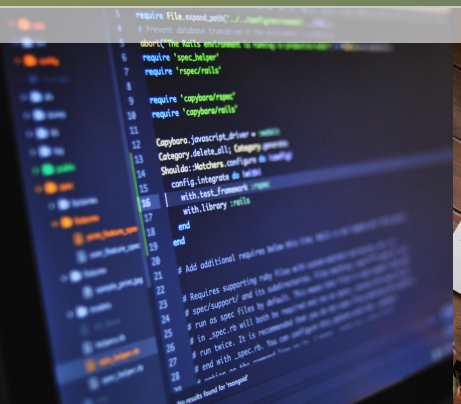
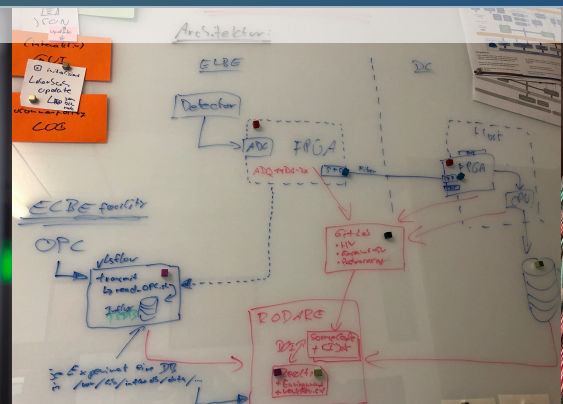
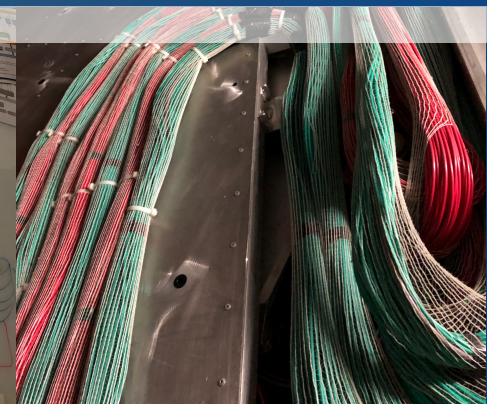
Dr.-Ing. Oliver Knodel

Übungsaufgaben — Hardwaretechnik

Dresden // April & Mai, 2024



Aufgaben zur Informationsdarstellung

1. Was ist ein Alphabet?
2. Was ist ein Maschinenwort und in welchen Zusammenhang damit stehen die Formate Nibble, Halbwort, Wort, Doppel- und Quadwort?
3. (Optional) Was ist ein:
   a) Binärcode
   b) BCD-Code
   c) 1-aus-n-Code
   d) ASCII-Code

Aufgaben zur Informationsverarbeitung

1. Machen Sie sich mit den Konvertierungsverfahren vertraut: Beispiel: $4123{,}625_{10}$
   a) **Dezimal – Dual**, **Dual – Dezimal**
   b) Dezimal – Oktal, Oktal – Dezimal
   c) Dezimal – Hexadezimal, Hexadezimal – Dezimal
2. Konvertieren Sie die Dualzahl aus 1. von Dual zu Hexadezimal zu Oktal
3. Konvertieren Sie $100101{,}011_2$ nach Dezimal
4. Darstellung im 2er-Komplement (4 Bit):
   a) -6_{10}
   b) $+5_{10}$
   c) -15_{10}

Digitale Logik und Schaltwerke

Aufgaben — Digitale Logik I

1. Gegeben sei die Schaltfunktion: $\mathbf{y = (x_2 + x_1)(x_1 + x_0)}$

a) Lösen Sie die Klammern auf und optimieren Sie die Gleichung.

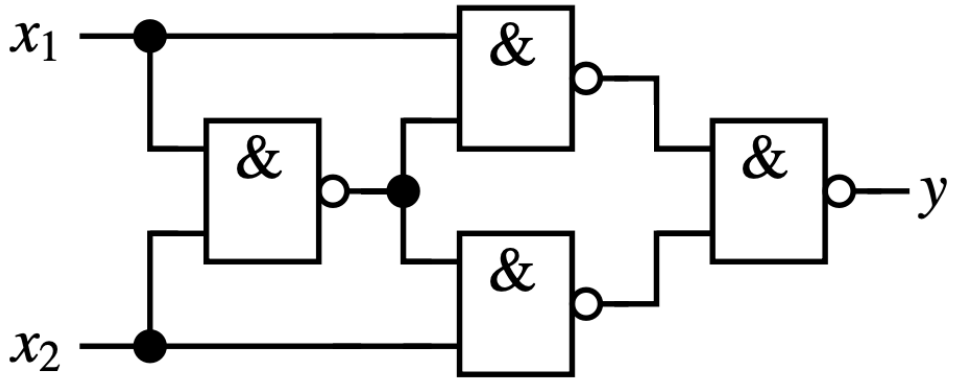
b) Stellen Sie die Funktion als Wertetabelle dar.

c) Realisieren Sie die Funktion durch eine logische Schaltung.

d) Formen Sie die Funktion so um, dass sie ausschließlich mit NAND-Gattern mit zwei Eingängen realisiert werden kann.

Aufgaben — Digitale Logik II

2. Das nachfolgende Schaltnetz ist zu analysieren:



a) Stellen Sie die vollständige Wertetabelle für die Ausgangsfunktion **y** auf (durch Simulation und symbolischer Berechnung der Schaltung).

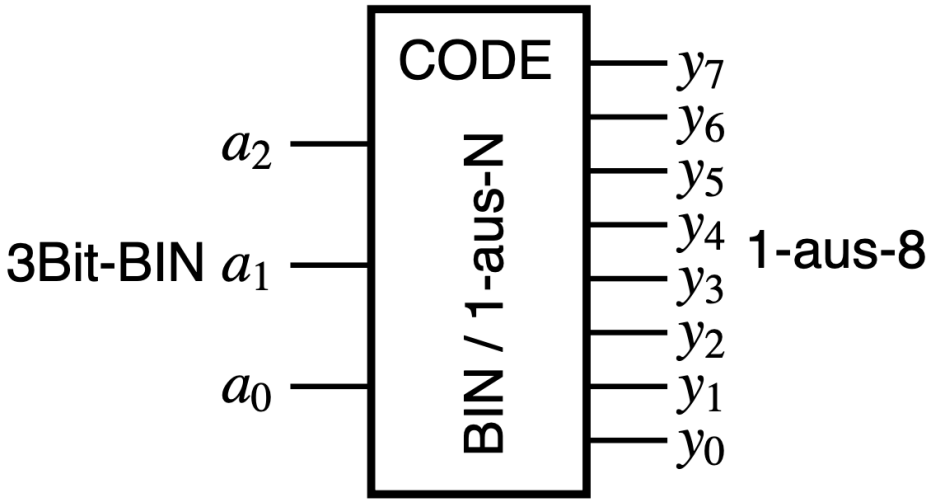
b) Als disjunktive Normalform (Oder-Verknüpung aller Vollkonjunktionen mit dem logischen Wert 1)

Aufgaben — Digitale Logik III

3. Der Aressdekodierer kodiert binäre Adressen in eine 1-aus-N Kodierung. Bei dieser Kodierung ist nur ein Ausgang entsprechend dem Eingangsbinäräquivalent mit 1 belegt.

a) Stellen Die die Wertetabelle auf.

b) Geben Sie die acht Funktionen für den Kodierer an.



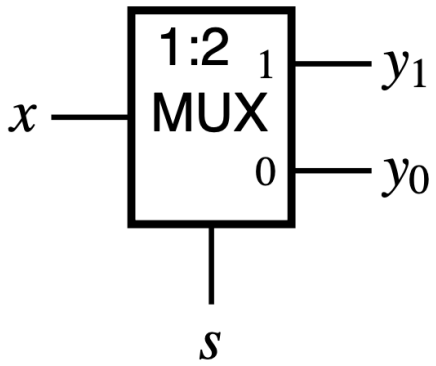
Aufgaben — Digitale Logik IV

4. 1:2 Multiplexer bzw. 2:1 Multiplexer (Demultiplexer) dienen zur Umschaltung von Datenleitungen. Zusammengefaßt als Multiplexerbäume können mit ihnen komplexe Schaltnetze realisiert werden.

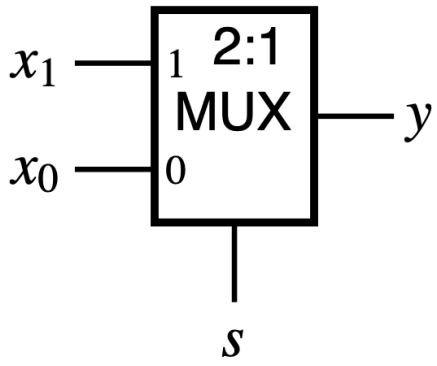
a) Stellen Sie die Wertetabellen auf.

b) Geben Sie die Funktionen y für beide Multiplexer und Demultiplexer an.

Multiplexer



Demultiplexer



Aufgaben — FlipFlops

5. Machen Sie sich mit der Funktionsweise eines NOR-Grund-FlipFlops vertraut.

a) Stellen Sie die Wertetabelle auf.

b) Was unterscheidet ein FlipFlop von einem Schaltnetz?

c) Stellen Sie die Zustandsübergangstabelle auf.

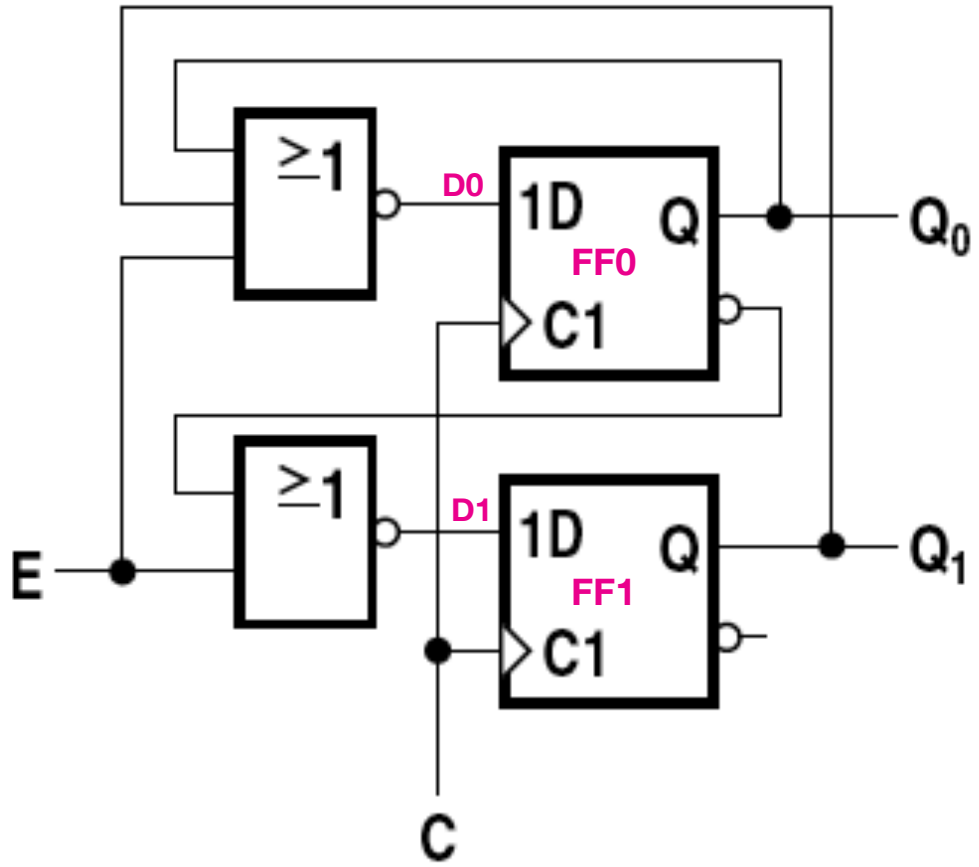
d) Was passiert bei der Eingangsbelegung R = S = 1

e) Entwerfen Sie eine kombinatorische Vor-Schaltung bei der das Setzen bei der Eingangsbelegung R = S = 1 dominiert.

f) Entwerfen Sie eine kombinatorische Vorschaltung, welche aus einem RS-FlipFlop ein D-FlipFlop macht.

Aufgaben — Synthese und Analyse von Schaltwerken I

6. Gegeben sei folgende Schaltung:



a) Ermitteln Sie die Ansteuerfunktionen für die FlipFlops.
b) Stellen Sie die Zustandsübergangstabelle auf.
c) Zeichnen Sie den Zustandsgraphen.
d) Geben Sie eine geeignete Bezeichnung für die Schaltung an.
e) Welcher Typ (Art der Ausgänge) liegt vor?

Aufgaben — Synthese und Analyse von Schaltwerken II

7. Zur Überwachung einer Signalleitung **x** soll ein Zustandsautomat eingesetzt werden. Der erwartete Signalverlauf ist dabei die strikt alternierende Folge von Nullen (0) und Einsen (1). Eine beliebige Abweichung von diesem Muster ist durch einen Impuls auf einem Fehlerindikationssignal **F** anzuzeigen.

   a) Wie wird eine Fehler identifiziert? Die unterstellte Fehlerzahl bei einer Abweichung soll möglichst gering sein. So weist etwa die Folge **0101000101** einen **Bitflip** von 1 nach 0 und nicht zwei Einschübe von 0 auf.

   b) Entwickeln Sie den Zustandsübergangsgraphen.

   c) Stellen Sie die Zustandsübergangstabelle auf. Es stehen D-FF mit einem gesonderten Rücksetzeingang zur Verfügung.

   d) Welcher Typ (Art der Ausgänge) liegt vor?

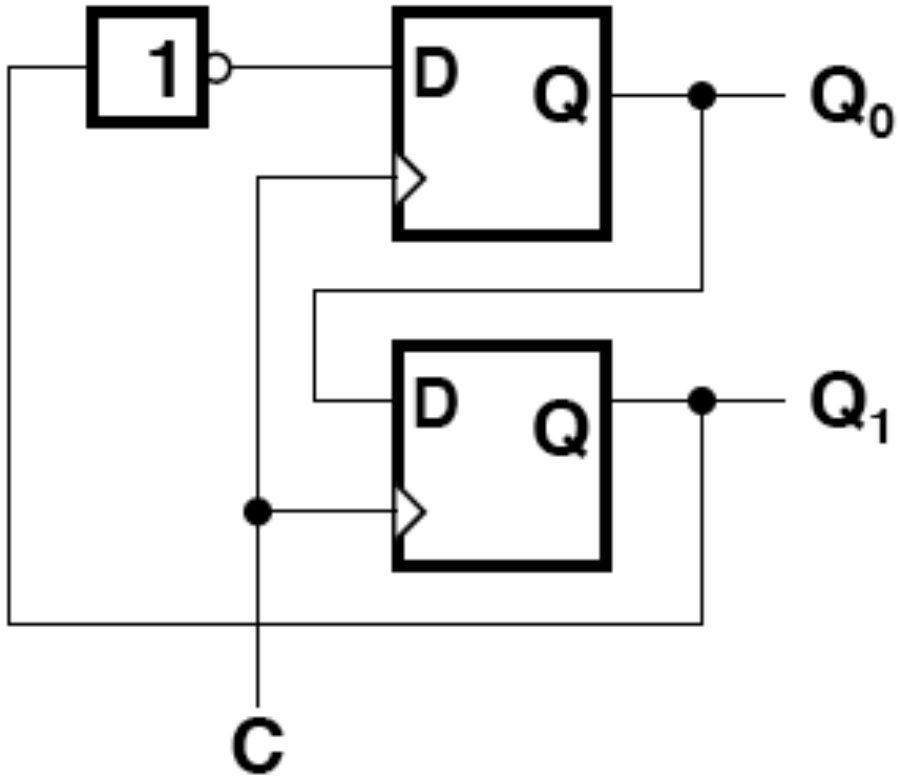
Aufgaben — Synthese und Analyse von Schaltwerken III

8. Ein getakteter serieller Datenbus **x** führt im Ruhezustand einen High-Pegel. Die Daten-übertragung wird durch einen Low-Impuls einer Taktlänge eingeleitet und umfasst jeweils genau die vier Bits der direkt darauf folgenden Takte.

   Entwerfen Sie einen den Zustandsgraphen für einen Automaten, der den Abschluss der Übertragung (**E**) eines solchen Nibbles und dann auch dessen Parität (**P**) signalisiert.

Aufgaben — Synthese und Analyse von Schaltwerken IV

9. Gegeben sei folgende Schaltung:



a) Ermitteln Sie die Ansteuerfunktionen für die FlipFlops.

b) Welcher Typ (Art der Ausgänge) liegt vor?

c) Stellen Sie die Zustandsübergangstabelle auf.

d) Zeichnen Sie den Zustandsgraphen.

e) Nach welchem Code zählt dieses Schaltwerk?

Architekturkonzepte

Aufgaben — Architekturkonzepte in der Rechnerarchitektur I

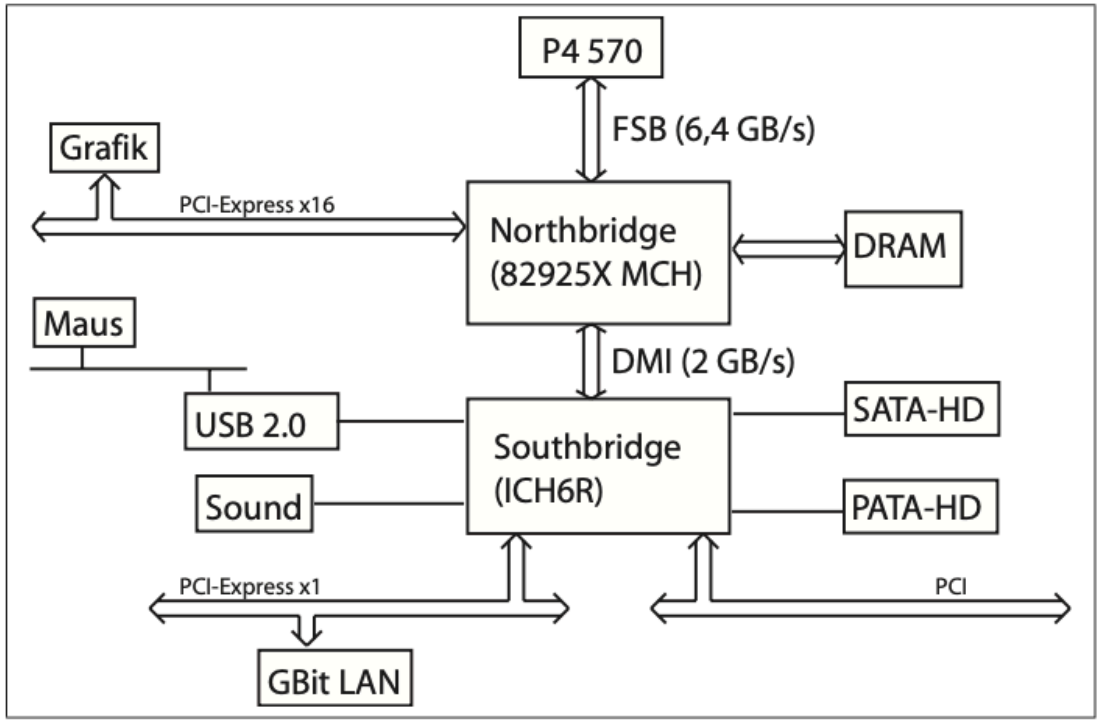
1. Was ist das Besondere an der Harvard Architektur und wie unterscheidet sie sich von der Princeton Architektur?
2. Wie unterscheiden sich Kontroll- und Datenflußarchitekturen voneinander?
3. Wie sind moderne Prozessoren in die unterschiedlichen Klassifikationen (Flynn, Giloi, Erlangen) und Architekturen (Princeton, Harvard) einzuordnen?
4. Ein Multiprozessorsystem mit 4 Prozessoren, pro Prozessor ein Rechenwerk mit 32 Bit Verarbeitungsbreite und 5-fachem Phasenpipelining ist in ECS zu beschreiben.
5. Beschreiben Sie ein (mögliches) Multiprozessorsystem mittels PMS.

Aufgaben — Architekturkonzepte in der Rechnerarchitektur II

6. Stellen Sie die Prozessorarchitektur eines Pentium-4 Prozessors in PMS dar. Die Architektur kann an folgendem vereinfachten Prinzip zusammengefasst werden:
    - Die Bus Interface Unit stellt die Verbindung von Level-2-Cache und dem externen FrontSideBus dar.
    - Eine Instruction-Fetch-Unit führt die Befehle aus dem L2-Cache der Prozessorpipeline zu.
    - Für die Operanden und Ergebnisse steht der Level-1-Cache zur Verfügung, der seine Daten ebenfalls dem L2-Cache entnimmt.
    - Zusätzlich besitzt der Prozessor einen Trace Execution Cache.

Aufgaben — Architekturkonzepte in der Rechnerarchitektur II

7. Beschreiben Sie das externe Pentium-4-System mittels der grafischen PMS-Darstellung. Zum Einsatz kommt ein Intel Pentium 4 570 (3,8 GHz) auf einem Mainboard mit Intel 82925X Chipsatz. Verwenden Sie das abgebildete Blockdiagramm als Grundlage für Ihre Beschreibung.



Aufgaben — Konzepte des Von-Neumann Rechners

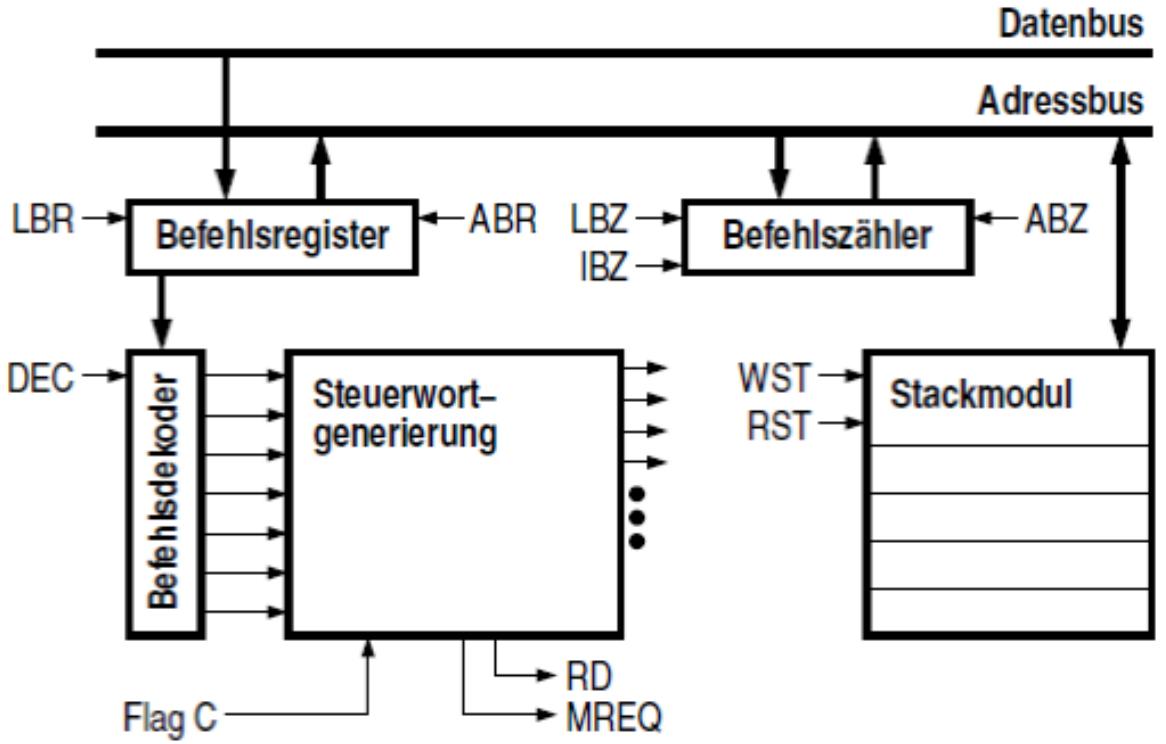
1. Aus welche Komponenten/Struktureinheiten besteht die von-Neumann Architektur? Was sind die wesentlichen Merkmale und Engpässe?
2. Ein Von-Neumann Rechner ist mittels PMS und ECS zu beschreiben.
3. Wie ist ein Befehlswort aufgebaut?
4. Wie sind die Aufgaben der Befehlsabarbeitung zwischen Steuer- und Rechenwerk verteilt?
5. Was ist die Aufgabe der zentralen Steuerschleife?

Aufgaben — Steuerwerk I

1. Zählen Sie die Grundkomponenten eines Steuerwerks auf und benennen Sie kurz den Funktion.
2. Welche drei wesentlichen Phasen durchläuft die Abarbeitung eines Befehles?

Aufgaben — Steuerwerk II

3. Gegeben ist ein Steuerwerk mit folgender Grundstruktur



Der Stapelspeicher zur Verwaltung des Aufrufstacks ist in Hardware direkt im Steuerwerk realisiert.

Aufgaben — Steuerwerk III

Folgende Steuersignale sind relevant:

`LBR`	Befehlsregister vom Datenbus laden
`ABR`	Adressteil des Befehlsregisters auf Adressbus legen
`LBZ`	Befehlszähler vom Adressbus laden
`ABZ`	Befehlszähler auf Adressbus ausgeben
`IBZ`	Befehlszähler inkrementieren
`WST`	Datum vom Adressbus auf den Stack legen
`RST`	Oberstes Stackelement vom Stack nehmen und auf Adressbus ausgeben
`DEC`	Befehl aus dem Befehlsregister dekodieren
`MREQ`	Hauptspeicher anfordern
`RD`	Hauptspeicher lesen

Der Lesezyklus des Hauptspeichers benötigt drei Takte:

1. Adresse anlegen, `MREQ` und `RD` aktivieren.
2. Wartetakt.
3. Daten auf Datenbus.

Entwerfen Sie eine Schaltung zur Ablaufsteuerung der folgenden Befehle:

- unbedingter Sprungbefehl (`JMP`),
- bedingter Sprungbefehl (`JC` für ein einziges Bedingungsflag *C*),
- unbedingter Unterprogrammaufruf (`CALL`), und
- Unterprogrammrückkkehr (`RET`)!

Aufgaben — Programmablauf

1. Gegeben sei das folgende Maschinenprogramm mit Sprüngen (`JMP`), Unterprogrammaufrufen (`CALL`), Rücksprüngen (`RETURN`) und abstrahierten Verarbeitungsbefehlen (`VBx`):

Adresse	Befehl
0x200	VB1
	VB2
	CALL @1
	VB3
	JMP @3
@1:	VB4
	CALL @4
@2:	VB5
	RETURN
@3:	VB6
	JMP @5
@4:	VB7
	JMP @2
@5:	VB8

Das Programm liegt im Speicher ab der Adresse `0x200`. Sprünge und Unterprogrammaufrufe belegen zwei Speicherzellen, alle anderen Befehle eine. Unterprogrammaufrufe hinterlegen die Rücksprungadresse auf einem Stack, von dem diese bei einem Rücksprung wieder in den Befehlszähler geladen wird.

(a) Ordnen Sie jedem Befehl seine Adresse im Speicher zu!

(b) Protokollieren Sie einen Programmdurchlauf! Geben Sie dazu jeweils den Inhalt des Befehlszählers *vor* der Befehlsausführung, die Befehlsmnemonik des ausgeführten Befehls und den Stackinhalt *nach* dessen Ausführung an!

Aufgaben ISA I — Überblick Architekturen

1. Was ist ein Befehlssatz?
2. Was unterscheidet CISC und RISC voneinander?
3. Was sind die Unterschiede zwischen Stack-, Akkumulator, und Universalregister-Architektur?
4. Was sind 2-, und 3-Adressmaschinen?

Aufgaben ISA II — Adressmaschine

5. Folgendes Programm für eine 3-Adressmaschine ist gegeben:

```
ADD     R3,R1,R2
SUB     R4,R1,R2
MUL     R1,R3,R4
MUL     R3,R1,R1
MUL     R3,R3,R1
```

Assemblersyntax: `<Operation><Ziel><Quelle1><Quelle2>`

a) Übersetzung auf eine 2-Adressmaschine?
b) Übersetzung auf eine Akkumulator-Architektur?
c) Ergänzung der notwendigen Anweisungen bei der 2- und 3-Adressmaschine, wenn alle Operanden zuerst aus dem Hauptspeicher geladen werden müssen und das Ergebnis dort wieder abgelegt wird. Anzahl der notwendigen Befehle?

Aufgaben ISA III — Stackarchitektur

6. Eine 8-Bit-Stackarchitektur mit 12-Bit-Adressraum ist gegeben:

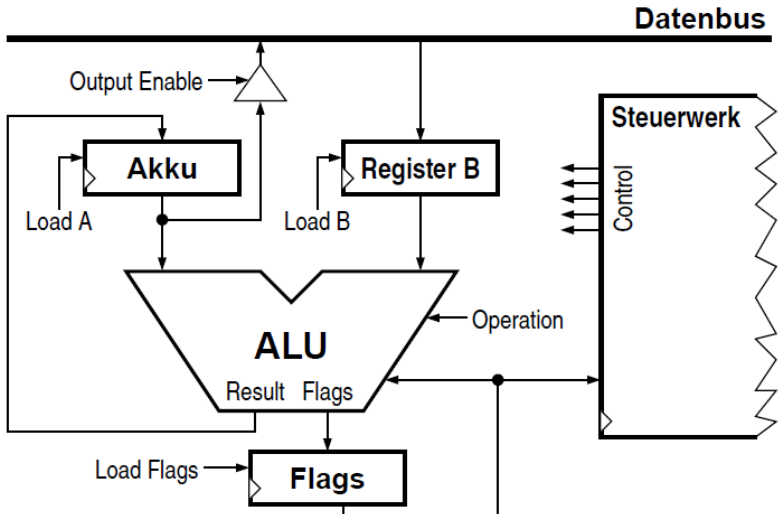
Mnemonik	Maschinencode	Beschreibung
`NOP`	0000 0000	Tue nichts.
`POP`	0001 0000	Verwerfe oberstes Stackelement.
`AND`	0001 0001	Nehme oberen beiden Werte vom Stack, UND-verknüpfe sie bitweise und lege das Ergebnis wieder auf den Stack.
`OR`	0001 0010	Nehme oberen beiden Werte vom Stack, ODER-verknüpfe sie bitweise und lege das Ergebnis wieder auf den Stack.
`XOR`	0001 0011	Nehme oberen beiden Werte vom Stack, XOR-verknüpfe sie bitweise und lege das Ergebnis wieder auf den Stack.
`ADD`	0001 0100	Nehme oberen beiden Werte vom Stack, addiere sie und lege das Ergebnis wieder auf den Stack.
`PUSH #i`	1100 <i>	Lege vorzeichenerweiterte 4-Bit-Konstante *i* auf den Stack.
`DUP`	1101 1101	Dupliziere obersten Stackwert noch einmal auf den Stack.
`PUSH [A]`	1110 <A.hi>, <A.lo>	Kopiere Speicherinhalt von Adresse *A* auf den Stack.
`POP [A]`	1111 <A.hi>, <A.lo>	Verschiebe obersten Stackwert an die Speicheradresse *A*.

(a) Wie lassen sich das Einer- bzw. das Zweierkomplett des obersten Stackelementes x bilden? Geben Sie die erforderlichen Befehlssequenzen in Mnemoniken und im hexadezimalen Maschinencode an! Dokumentieren Sie den Stackzustand!

(b) An den Speicheradressen `0x230 ...0x232` liegen die Werte der Variablen a, b und c in dieser Reihenfolge. Geben Sie wie oben ein Programm zur Berechnung von $y = 3(a - b) - c$ an! Das Ergebnis y soll abschließend an der Adresse `0x240` abgelegt werden.

Aufgaben ISA IV — Akkumulatorarchitektur I

7. Gegeben ist folgendes Rechenwerk mit Akkumulator:



(a) Wie viele Operandenadressen enthält ein typischer arithmetischer Befehl einer solchen Akkumulatorarchitektur?

(b) Auf die Angabe welcher Adressen kann durch welche Adressierungsmodi verzichtet werden?

(c) Beschreiben Sie den zeitlichen Ablauf der Ausführung des Befehls `ADD [0x200]` *im Rechenwerk*! Der Speicher führt gerade den Speicherzugriff aus, der Speicheroperand liegt auf dem Datenbus. Welche Operation führt die ALU aus?

(d) Wie sieht im Vergleich dazu der Ablauf für den Befehl `LOAD [0x240]` aus?

(e) Wozu verwendet das Steuerwerk die im Rechenwerk gebildeten Flags?

Aufgaben Stack

1. Ein Stack wird in der Regel durch eine Speicheradresse repräsentierten Stackpointer (SP) realisiert.
    a) Welche beiden naheliegenden Alternativen Speicherplätze kann der Stackpointer identifizieren?
    b) Wie ist der Stackpointer bei den Operationen `PUSH` und `POP` zu korrigieren?
2. Ein Stack kann auch als Aufrufstack eine Hierarchie von Unterprogrammaufrufen realisieren.
    a) Wie kann auf einem Stack weiterer Speicher für lokale Variablen reserviert werden?
    b) Über welchen Adressierungsmodus kann auf die so alliierten Variablen zugegriffen werden?
    c) Wie können die Variablen wieder frei gegeben werden?
    d) Für welches Aufrufmuster in einem Programm ist eine derartige Vorgehensweise zwingend erforderlich?

Aufgabe — ALU

1. Eine einfache ALU ist mit Logikgattern zu entwerfen:
    a) Nutzen Sie zuerst mehrere Volladdierer (VA) zur Realisierung der Addition zweier 4-Bit Operanden.
    b) Erweitern Sie den 4-Bit Addierer zu einer 4-Bit ALU, welche auch eine Subtraktion ausführen kann. Über das Steuersignal **`command = add/sub`** soll die entsprechende Operation ausgeführt werden.
    c) Fügen Sie der Schaltung ein Statusregister mit den Flags **V**, **C** und **S** hinzu.
2. Entwerfen Sie ausgehend vom schriftlichen Multiplikationsschema eine Schaltung die zwei vorzeichenlose 4-Bit Zahlen multipliziert:
    a) Wie funktioniert die schriftliche Multiplikation Binärer Zahlen? Wie viele Bits kann das vollständige Produkt umfassen?
    b) Entwerfen Sie eine geeignete Multipliziererzelle (MC), die die Operation für eine Ziffer in der Produktmatrix realisiert.
    c) Entwerfen Sie einen kompletten 4 × 4 Feldmultiplizierer unter Nutzung dieser Multipliziererzelle.

Aufgabe — ALU

3. Gegeben sei eine **8-Bit**-Akkumulator-Architektur. Das Rechenwerk arbeitet im 2er-Komplement und bildet die üblichen Flags (Zero, Sign, Carry, Overflow).

    a) Die Ergebnisse und die Flags nach der Addition folgender Wertpaare mit Hilfe des ADD-Befehls sind zu berechnen:

        i. $27_{16} + 6C_{16}$
        ii. $E6_{16} + 1C_{16}$
        iii. $43_{16} + 84_{16}$
        iv. $9A_{16} + 66_{16}$

    b) Wann sind Ergebnisse in vorzeichenloser oder in vorzeichen-behafteter Interpretation von einer Bereichsüberschreitung betroffen? Wie kann das an den Flags abgelesen werden?

Aufgaben — ALU

4. Gegeben sei eine **8-Bit-Akkumulator-Architektur**, die über einen byteweise adressierten Speicher mit 10 Bit Adressraum verfügt. Das Rechenwerk arbeitet im 2er-Komplement und bildet die üblichen Flags. Datenworte werden in **Big Endian** abgelegt.

Mnem.	Maschinencode	Beschreibung	Aktualisierte Flags
JPf L	00<f:2> <0:4>	Bedingter Sprung bei gesetztem Bedingungsflag.	—
JNf L	01<f:2> <0:4>	Bedingter Sprung bei nicht gesetztem Bedingungsflag.	—
JMP L	1000 00<0.hi:2>, <0.lo:8>	Unbedingter Sprung.	—

- Die Assemblermnemoniken verwenden benannte Sprungmarken (Label).
- Die Sprungziele sind PC-relativ und bestimmen sich aus dem im 2er-Komplement vorliegenden, vorzeichenbehafteten Sprungoffset *O* relativ zur Adresse des regulären Folgebefehls.
- Das den Sprung bedingende Flag ist wie folgt kodiert:

Z	00	–	Zero	Null als Ergebnis.
C	01	–	Carry	Auslaufender Übertrag.
V	10	–	Overflow	Arithmetischer Überlauf (vorzeichenbehaftet).
S	11	–	Sign	Negatives Ergebnis.

NOT	1000 0100	A←$\bar{A}$	Invertiere Akkumulator bitweise.	*Z, S*
INC	1000 0110	A←A+1	Inkrementiere Akkumulator.	*Z, C, V, S*
LD [P]	1100 00<P.hi:2>, <P.lo:8>	A←<P>	Lade Akkumulator mit Speicherwert an Adresse P.	—
ST [P]	1100 01<P.hi:2>, <P.lo:8>	<P>←A	Kopiere Akkumulator an Speicherplatz mit Adresse P.	—
ADD [P]	1100 10<P.hi:2>, <P.lo:8>	A←A+<P>	Addiere Speicherwert an Adresse P zum Akkumulator.	*Z, C, V, S*
ADC [P]	1100 11<P.hi:2>, <P.lo:8>	A←A+<P>+C	Addiere Speicherwert und Carry-Flag zum Akkumulator.	*Z, C, V, S*

Aufgaben — ALU

Fortsetzung Aufgabe 4:

a) Wie groß ist der maximal adressierbare Speicherbereich?

b) Disassemblieren Sie den Maschinencodebefehl 00_{16}. Welche Auswirkungen hat er? Welche alternative Bezeichnung für den Befehl bietet sich an?

c) Die **16-Bit** Variablen a und b liegen ab der Adresse 0x200, bzw. 0x202 im Speicher. Schreiben Sie ein Programm, das $c = a + b$ berechnet und das Ergebnis c im Speicher auf Adresse 0x204 ablegt. Geben Sie sowohl den Befehl als auch den hexadezimalen Maschinencode an.

d) Wie kann der Befehl ADC [0x202] durch andere Befehle abgebildet werden, wenn er nicht zur Verfügung steht?

e) Erweitern Sie die Lösung so, dass $c = |a + b|$ berechnet wird.

Speicher und Cache

Aufgaben Alignment I

1. Gegeben ist folgender Speicherauszug aus einem byteweise addressierbaren Speicher:

Adresse	Inhalt
0x0FD	0x02
0x0FE	0xD5
0x0FF	0x68
0x100	0x4F
0x101	0x73
0x102	0xC9
0x103	0x1B

(a) Welche Werte werden bei folgenden Zugriffen von der Speicheradresse 0x100 gelesen?

Endian	Byte	16-Bit-Wort	32-Bit-Wort
Little			
Big			

(b) An welcher Adresse befindet sich, je nach Endian, das niederwertigste Byte des 32-Bit-Wortes an Adresse 0x100?

Aufgaben Alignment II

2. Ein Record besteht aus mehreren Datenfeldern für Ganzzahlen unterschiedlicher Bitbreite:

```
record
  a : int16;      // 16 Bit
  b : int8;       //  8 Bit
  c : int16;      // 16 Bit
  d : int32;      // 32 Bit
end record;
```

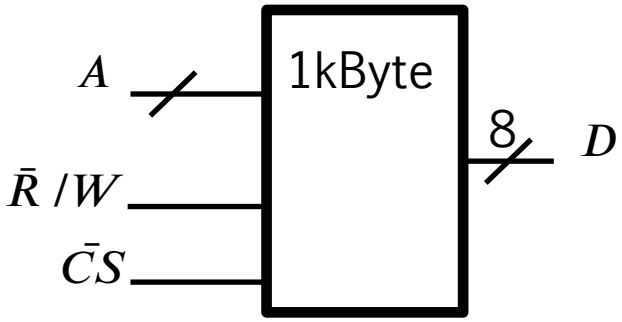
In einem Programm werde nun eine Instanz dieses Records ab der Adresse `0xA38` im byteadressierten Speicher mit {`.a=0x341A,.b=0x4D,.c=0x90F2,.d=0xA37B672E`} initialisiert. Zeigen Sie den Auszug des belegten Speichers nach dessen Initialisierung, wenn:

(a) das Record dicht in Little-Endian gepackt wird, und

(b) jedes Datenfeld des Records (ohne Umordnung!) gemäß seiner Größe *aligned* (= *natürliches Alignment*) in Big-Endian abgelegt wird!

Aufgabe – Speicherdimensionierung

1. Folgende Speicherelemente mit der Größe von 1kByte sind gegeben. Der Datenbus (D) hat eine Breite von 8 Bit:



a) Wie viele Adressbits (A) hat ein solches Element?

b) Wie kann ein Speicher von 2 kByte mit einem Datenbus von 16 Bit Breite realisiert werden?

c) Wie kann der Adressraum des Speichers (unter Beibehaltung des Datenbusses von 8 Bit) auf 4 kByte erhöht werden?

Aufgaben – Cache I

1. Stellen Sie eine moderne Cache-Architektur mit drei Ebenen und Unterteilung der L1-Ebene in Harvard mittels PMS dar.
2. Das Speichersubsystem ist in modernen Rechnen hierarchisch aufgebaut.
    a) In welchen markanten Eigenschaften unterscheiden sich die Komponenten dieser Hierarchie?
    b) Welche typische Eigenschaft des Verlaufs von Speicherzugriffen wird von der Speicherhierarchie zur Leistungssteigerung ausgenutzt? Benennen Sie deren beide Erscheinungsformen und beschreiben Sie diese kurz.

Aufgaben – Cache II

3. Für eine byteorientierte Speicherhierarchie mit 32-Bit-Adressen werde ein vollassoziativer Cache mit einer Kapazität von 16 KiByte angenommen. Seine Cachezeilen sind 64 Byte lang und in Worte von je 4 Byte untergliedert.
    a) Wie wird eine Speicheradresse zum Zweck eines Cache-Lookups untergliedert?
    b) Wie viele Zeilen enthält der Cache?
    c) Welche Kapazität hat der Tag-Speicher (ohne etwaige Statusbits)?
    d) Was muss für einen Cache-Hit neben einem erfolgreichen Tag-Abgleich noch überprüft werden?

Aufgaben – Cache III

4. Wie ändern sich die Parameter, wenn der Cache aus Aufgabe 3 als Direkt Abgebildeter Cache vorliegt?
5. Ein byteorientiertes Speichersystem mit einem physischen 16-Bit Adressraum enthält einen 2-fach satzassoziativen Cache mit 16 Cachezeilen über je 16 Byte. Die Ersetzungsstrategie ist LRU (Least-recently-used).
    a) Geben Sie die Adressaufteilung an.
    b) Über den leeren Cache laufen nun folgende Speicherzugriffe:

Nr..	R/W	Adresse	Index	Tag	Hit	Ersetzt Nr.
1 .	RD	0xA31B	0x1	0x146	–	–
2 .	RD	0xA39F				
3 .	WR	0xA3D0				
4 .	RD	0xA3D4				
5 .	RD	0x079C				
6 .	RD	0xA315				
7 .	WR	0xA3D7				

Aufgaben – Cache IV

6) In einem Rechner mit einem Hauptspeicher von 4 GByte wird eine Programmschleife durchlaufen, welche auf dem Adressbereich von `0xA010 0000` bis `0xA010 01FF` liegt. Die Befehle liegen im 32-Bit Format vor und der Prozessor besitzt als Befehlscache einen Direkt-Abgebildeten Cache, bei dem jede Cacheline aus 4 Byte besteht und der Index 6 Bit breit ist.
    a) Welche Adressbits bilden den Tag?
    b) Wieviel Zeilen hat der Cache?
    c) Wie groß ist der Cache?
    d) Wie viele Bytes umfasst die Programmschleife?
    e) Wie hoch ist die Trefferquote des Cache?

Pipelining

Aufgaben Pipelining

1. Gegeben sei ein RISC-Prozessor mit Harvardarchitektur, 16 Universalregistern und einer vierstufigen Pipeline:
    1. IF instruction fetch
    2. ID/OF instruction decode, operand fetch
    3. EX/LS execute, load/store
    4. WB write back

    Der Prozessor verfügt über kein weiteres Bypassing (keine Forwarding-Register).

    Auf diesem Prozessor soll nun folgendes Maschinenprogramm abgearbeitet werden. Die Besonderheiten der Befehlsabarbeitung sind dabei noch nicht berücksichtigt:

```
...
1.    M1:   SUB R3, R1, R2   (R3 <- R1 - R2)
2.          ADD R4, R6, R3   (R4 <- R6 + R3)
3.          SUB R5, R1, #1   (R5 <- R1 - 1)
4.          ADD R6, R1, R5   (R6 <- R1 + R5)
5.          LDI R6, #2       (R6 <- 2)
...
```

    a) Eine wieviel-Adressmaschine liegt vor?
    b) Geben Sie das Pipeline-Taktschema des unmodifizierten Programms an!
    c) Identifizieren Sie die darin auftretenden Pipeline-Konflikte!
    d) Wie können diese Konflikte gelöst werden, wenn der Prozessor diese nicht selbständig erkennt und behebt?


END