KIT-Fakultät für Informatik
Prof. Dr. Mehdi Tahoori, Prof. Dr. Wolfgang Karl

Aufgabenblätter zur Prüfung

Digitaltechnik und Entwurfsverfahren (TI-1)

und

Rechnerorganisation (TI-2)

am 29. März 2023, 11:00 – 13:00 Uhr

- Beschriften Sie bitte gleich zu Beginn jedes Lösungsblatt deutlich lesbar mit Ihrem Namen und Ihrer Matrikelnummer.
- Diese Aufgabenblätter werden nicht abgegeben. Tragen Sie Ihre Lösung deshalb ausschließlich in die für jede Aufgabe vorgesehenen Bereiche der Lösungsblätter ein. Lösungen auf separat abgegebenen Blättern werden nicht gewertet.
- Außer Schreibmaterial sind während der Klausur keine Hilfsmittel zugelassen. Täuschungsversuche durch Verwendung unzulässiger Hilfsmittel führen unmittelbar zum Ausschluss von der Klausur und zur Note „nicht bestanden“.
- Soweit in der Aufgabenstellung nichts anderes angegeben ist, tragen Sie in die Lösungsblätter bitte nur Endergebnisse und Rechenweg ein. Die Rückseiten der Aufgabenblätter können Sie als Konzeptpapier verwenden. Weiteres Konzeptpapier können Sie auf Anfrage während der Klausur erhalten.
- Halten Sie Begründungen oder Erklärungen so kurz und präzise wie möglich. Der auf den Lösungsblättern für eine Aufgabe vorgesehene Platz lässt nicht auf den Umfang einer korrekten Lösung schließen.
- Die Gesamtpunktzahl beträgt 90 Punkte. Zum Bestehen der Klausur sind mindestens 40 Punkte zu erreichen.

Viel Erfolg und viel Glück!

Aufgabe 1 *Minimierungsverfahren* (9 Punkte)

Eine vollständig definierte Schaltfunktion $y = f(d, c, b, a)$ ist gegeben durch

$$y = \text{MAXt}(0, 1, 4, 5, 8, 10, 12, 14).$$

1. Tragen Sie die Funktion f in das KV-Diagramm im Lösungsblatt ein. Zeichnen Sie *alle* Prim-Nullblöcke klar und eindeutig ein und geben Sie die zugehörigen Primimplikate an. 3 P.

2. Geben Sie *alle* konjunktiven Minimalformen (KMF) von f an. 1 P.

3. Geben die kürzeste Gleichung für f an, die als Ausgangspunkt für die Gewinnung *aller* Primimplikanten mit Hilfe des Nelson-Verfahrens geeignet ist. 1 P.

Eine weitere, vollständig definierte Schaltfunktion $z = g(d, c, b, a)$ ist gegeben durch den Würfel ihrer Einsstellen $\mathcal{C}_1$. Die Variablenreihenfolge im Würfel ist d, c, b, a.

$$\mathcal{C}_1 = \{1, 1, -, 1), (1, 0, -, 1), (1, 0, -, 0), (0, 1, 1, -), (0, 0, 1, 1)\}$$

4. Bestimmen Sie mit Hilfe des Consensus-Verfahrens die Menge aller Primimplikanten. Die prinzipielle Vorgehensweise bei der Anwendung des Consensus-Verfahrens soll aus der Lösung ersichtlich sein. Verwenden Sie hierzu die im Lösungsblatt vorbereitete Tabelle. 4 P.

Aufgabe 2 *Schaltnetze und CMOS-Technologie*(10 Punkte)

Gegeben sei das in Abbildung 1 dargestellte Schaltnetz der Schaltfunktion $y = f(d, c, b, a)$:

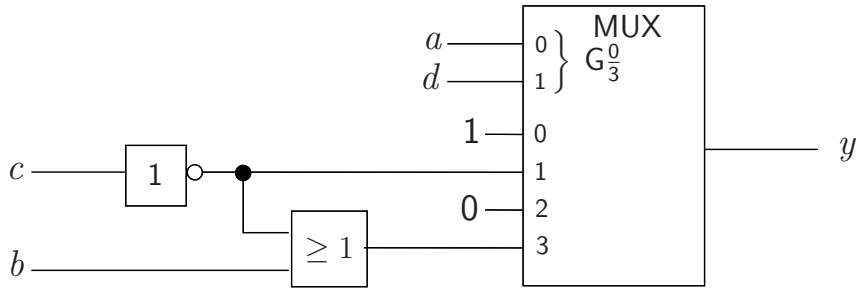


Abbildung 1: Schaltnetz der Funktion $y = f(d, c, b, a)$

1. Geben Sie die Schaltfunktion $y = f(d, c, b, a)$ an. 1 P.

2. Formen Sie die Schaltfunktion y mit Hilfe der Regeln der Schaltalgebra in eine Minimalform um. 1 P.

3. Die Schaltfunktion y soll unter ausschließlicher Verwendung von NAND-Gattern realisiert werden. Formen Sie die im letzten Aufgabenteil gefundene Minimalform entsprechend um und zeichnen Sie das zugehörige Schaltnetz. Die Eingangsvariablen stehen nur nicht negiert zur Verfügung. 3 P.

4. Die Schaltfunktion 2 P.

$$g \;=\; \mathrm{NAND}_3(x_1, x_2, x_3) \;=\; \overline{x_1 \;\wedge\; x_2 \;\wedge x_3}$$

soll unter ausschließlicher Verwendung von NAND-Gattern mit zwei Eingängen realisiert werden. Wandeln Sie die Schaltfunktion entsprechend um. Zeichnen Sie das Schaltbild.

5. Geben Sie eine CMOS-Realisierung des in Abbildung 2 dargestellten 2:1-Multiplexers an. Es stehen Ihnen dabei zwei Transmission-Gates und ein Inverter zur Verfügung. Geben Sie ein vollständiges CMOS-Schaltbild inklusive der CMOS-Darstellungen für Inverter und Transmission-Gates an. 3 P.

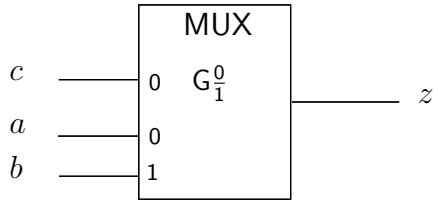


Abbildung 2: 2:1-Multiplexer

Aufgabe 3 *Laufzeiteffekte* (5 Punkte)

Eine Schaltfunktion $y = f(c, b, a)$ sei durch das Schaltnetz in Abbildung 3 mit den angegebenen Verzögerungszeiten realisiert. Betrachten Sie den **im Lösungsblatt** angegebenen zeitlichen Verlauf der Eingangsvariablen. Zu Beginn liegen alle Eingabevariablen stabil an.

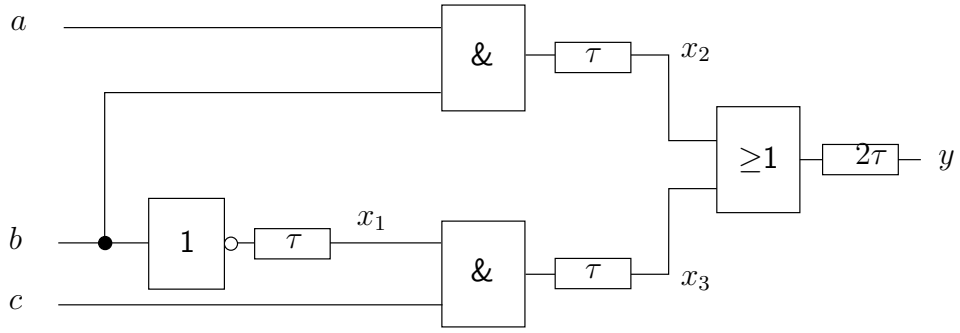


Abbildung 3: Schaltnetz der Schaltfunktion $y = f(c, b, a)$

1. Zeigen Sie anhand eines Zeitdiagramms, ob die folgenden Eingabewechsel einen Hasardfehler auslösen. Die Variablenreihenfolge sei (c, b, a). Es reicht aus, das Zeitdiagramm vollständig auszufüllen. 3 P.

   (a) b wechselt auf 1, d. h. Übergang $B_5 \rightarrow B_7$ zum Zeitpunkt t_0

   (b) b wechselt auf 0 zurück, d. h. Übergang $B_7 \rightarrow B_5$ zum Zeitpunkt t_1

2. Falls Sie Hasardfehler im letzten Aufgabenteil gefunden haben, dann geben Sie an, um welchen Typ von Hasardfehlern es sich handelt und wie sie behoben werden könnten. Geben Sie hierfür einen konkreten Lösungsvorschlag an. 2 P.

Aufgabe 4 *Schaltwerke* (12 Punkte)

1. Gegeben ist das in Abbildung 4 dargestellte Schaltwerk.

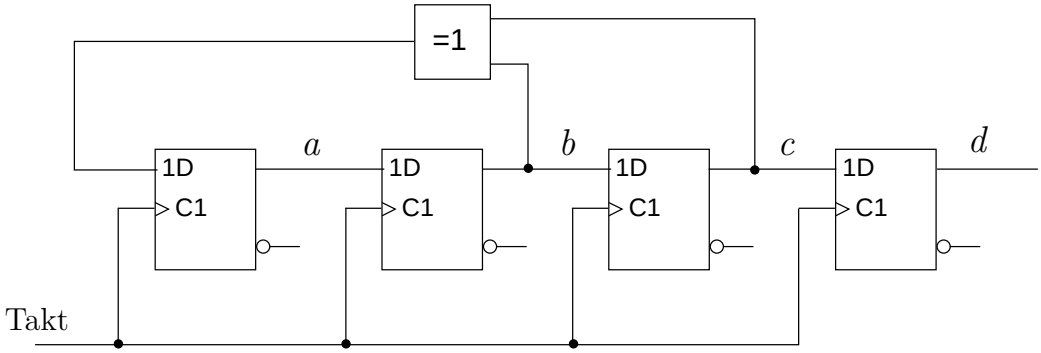


Abbildung 4: Schaltwerk

   (a) Ist das Schaltwerk synchron oder asynchron? 1 P.

   (b) Wie viele Zustände kann das Schaltwerk maximal annehmen? 1 P.

   (c) Vervollständigen Sie die Verläufe der Signale a, b, c und d im Lösungsblatt. 4 P.

2. Es soll ein synchroner modulo-8-Rückwärtszähler mit flankengesteuerten T-Flipflops entworfen werden.

   (a) Geben Sie den Automatengraphen des Zählers an. 2 P.

   (b) Stellen Sie die kodierte Ablauftabelle des Zählers auf. Verwenden Sie hierzu die im Lösungsblatt vorbereitete Tabelle. Die Zustände des Zählers seien mit Hilfe der Zustandsvariablen q_2, q_1 und q_0 dual kodiert. 1 P.

   (c) Geben Sie die Ansteuerfunktionen der verwendeten Flipflops in minimaler Form an. 1 P.

   (d) Zeichnen Sie das Schaltbild des Zählers. 2 P.

Aufgabe 5 *Rechnerarithmetik* (9 Punkte)

Hinweis: Geben Sie in dieser Aufgabe *immer* den Rechenweg an.

1. Wie viele Prüfbits sind für eine Einzelbit-Fehlerkorrektur in 200 Bit Datenwörtern erforderlich? 1 P.

2. Was ist der Unterschied zwischen einem *Carry-Ripple*-Addierer und einem *Carry-Lookahead*-Addierer? Wovon hängt die Additionszeit beim *Carry-Ripple*-Addierer ab? 2 P.

3. Wie viele Bits sind *mindestens* notwendig zur Darstellung der Zahl -70 als Zweierkomplementzahl? 2 P.

   - Geben Sie -70 als Zweierkomplementzahl mit minimaler Bitanzahl an.
   - Geben Sie -70 als 16-Bit-Zweierkomplementzahl an.

4. Gegeben sei die folgende 32 Bit Folge 4 P.

   1000 0011 0101 1000 0000 0000 0000 0101

   Was stellt diese Folge dar, wenn sie interpretiert wird als

   (a) BCD-Zahl.

   (b) Vorzeichenlose Dualzahl. Geben Sie den dezimalen Wert an.

   (c) Gleitkommazahl im IEEE-754-Standard in einfacher Genauigkeit. Geben Sie den dezimalen Wert an.

   Hinweis: Sie brauchen die Zweier-Potenzen nicht explizit auszurechnen.

Aufgabe 6 *RISC-V* (7 Punkte)

1. Welche Hardware-Komponenten sind für die Implementierung der Befehle vom R-Type bei RISC-V notwendig? Zeichnen sie die Komponenten mit ihren Verbindungen. 3 P.

2. Geben Sie für das folgende RISC-V Programmstück den Inhalt des Zielregisters in hexadezimaler Schreibweise nach der Ausführung des jeweiligen Befehls an. 4 P.

```
addi x1, zero, 0x69   # add immediate: rd = rs + imm
lui x2, 0x06          # load upper immediate: rd = imm << 12
andi x3, x1, 0x0a     # and immediate: rd = rs & imm
srai x4, x2, 8        # shift right arithemtic immediate: rd = rs >> imm
xor x5, x4, x3        # xor: rd = rs1 ^ rs2
slt x6, x5, x2        # set less than: rd = rs1 < rs2 ? 1 : 0
```

Aufgabe 7 *MIMA-Architektur* (5 Punkte)

Die MIMA ist die Ihnen aus der Vorlesung bekannte mikroprogrammierte Minimalmaschine (**siehe Beiblatt: Architektur der MIMA**), die nach dem von-Neumann-Prinzip aufgebaut ist, d. h. Maschinenbefehle werden sequentiell abgearbeitet. In der Lese-Phase wird ein über IAR adressierter Befehl aus dem Speicher gelesen und im IR abgelegt. Die Lese-Phase dauert 5 Taktzyklen. Im 6. Taktzyklus wird der Befehl dekodiert (Dekodier-Phase). Die Ausführungsphase beginnt im 7. Taktzyklus. Nach der Ausführung des Befehls folgt ein Zugriff auf den nächsten Befehl.

Nehmen Sie an, dass ein Hauptspeicherzugriff (Lesen und Schreiben) drei Takte dauert und währenddessen $\mathsf{R} = 1$ bzw. $\mathsf{W} = 1$ sein muss. Eine ALU-Operation sei nach einem Takt abgeschlossen.

Geben Sie das Mikroprogramm für die Lese-Phase (Fetch-Phase) in Register-Transfer-Schreibweise an, d. h. in der Form 5 P.

1. Takt: $\mathsf{IR} \rightarrow \mathsf{SAR};\quad \mathsf{R} = 1$
2. Takt:
3. $\vdots$

Aufgabe 8 *Cache-Speicher* (15 Punkte)

1. Bei einem Cache-Speicher mit einer Speicherkapazität von 128 KiByte ist die Hauptspeicheradresse in ein 17 Bit Tag-Feld, ein 12 Bit Index-Feld und ein 3 Bit Byte-Offset unterteilt. Geben Sie bei der Beantwortung der folgenden Fragen den Lösungsweg an.

   (a) Bestimmen Sie die Blockgröße in Bytes. 1 P.

   (b) Wie ist der Cache-Speicher organisiert? 2 P.

2. Es soll ein 2-fach-assoziativer Cache-Speicher *(2-way set associative cache)* mit 16 Sätzen und einer Blockgröße von 8 Byte realisiert werden. Nehmen Sie an, dass die Hauptspeicheradresse 32 Bit breit ist. Zur Verwaltung eines Cacheblocks wird nur ein Statusbit (*Valid*-Bit: V) verwendet. 3 P.

   Bestimmen Sie den insgesamt erforderlichen Speicherbedarf zur Realisierung dieses Cache-Speichers.

3. Gegeben sei ein direkt-abgebildeter Cache-Speicher (*direct mapped cache*) mit einer Speicherkapazität von 32 Byte und einer Blockgröße von 4 Byte. Als Aktualisierungsstrategie wird ein Durchschreibverfahren *(write through policy)* mit *write no-allocate* verwendet. Bei dieser Aktualisierungsstrategie wird ein CPU-Datum bei einem *Write Miss* nur in den Speicher geschrieben. Bei einem *Write Hit* wird ein CPU-Datum sowohl in den Cache als auch in den Speicher geschrieben. 4 P.

   Betrachten Sie die folgenden Lese- und Schreibzugriffe auf die in dezimaler Schreibweise angegebenen Adressen:

Adresse	**0**	**8**	**40**	**52**	**4**	**8**	**52**	**32**	**2**
read/write	r	r	w	r	r	r	w	w	r
Index	0								
Tag	0								
Byte-Offset	0								
Hit/Miss	Miss								

   Vervollständigen Sie die obige Tabelle im Lösungsblatt. Verwenden Sie dabei **Miss** für Cache-Miss und **Hit** für Cache-Hit.

4. Nachfolgend finden Sie eine Liste von Speicheradressen auf die sequentiell lesend zugegriffen werden soll. Es soll hierbei eine Byte-Adressierung verwendet werden.

   ```
   0x04, 0x34, 0xcf, 0x02, 0x4c, 0xcf, 0x84, 0xb6, 0xb5,
   0x07
   ```

   Geben Sie für jeden dieser Speicherzugriffe den Tag, den Zeilenindex und den Byteoffset in hexadezimaler Schreibweise an. Gehen Sie dabei von einem direkt-abgebildeten Cache *(direct-mapped cache)* mit **16 Speicherblöcken** aus, bestehend aus jeweils **vier Bytes**. Listen Sie ebenfalls auf, ob es sich dabei um einen Hit oder Miss beim Speicherzugriff handelt. Der Cache sei dabei initial leer. Verwenden Sie die auf dem Lösungsblatt bereit gestellte Tabelle. 5 P.

Aufgabe 9 *Virtuelle Speicherverwaltung* (6 Punkte)

Gegeben sei eine Speicherverwaltungseinheit (MMU) mit einer Seitengröße von 1 KiByte, 8 virtuellen Seiten und 4 physikalischen Seiten (Frames). Der aktuelle Ausschnitt der Seitentabelle ist in Tabelle 1 angegeben.

Virtuelle Seitennummer	Physikalische Seitennummer
0	3
1	1
2	-
3	-
4	2
5	-
6	0
7	-

Tabelle 1: Seitentabelle

1. Skizzieren Sie die Unterteilung der 32 Bit breiten virtuellen Adresse. 1 P.
2. Ermitteln Sie die physikalischen Adressen zu den folgenden virtuellen Adressen: 4 P.

$$1023,\ 1024,\ 4204,\ 6200$$

Zur Beschleunigung der Adressberechnung soll ein Cache-Speicher als *Translation-Lookaside-Buffer (TLB)* eingesetzt werden, der die letzten 32 Einträge aus dem Seitentabellen-Verzeichnis und der Seitentabelle speichert.

3. Wie breit ist der *Tag* eines Cache-Eintrags? Gehen Sie dabei von einer 32 Bit breiten virtuellen Adresse, einer 32 Bit breiten physikalischen Adresse und einer Seitengröße von 4 KiByte aus. 1 P.

Aufgabe 10 *Pipelining* (7 Punkte)

Das folgende Programmstück soll in der aus der Vorlesung bekannten fünfstufigen RISC-V -Pipeline ausgeführt werden.

```
S1:    lw    t1, 100(t0)
S2:    lw    t2, 104(t0)
S3:    add   t3, t2, t1
S4:    addi  t1, t2, 8
S5:    xori  t4, t0, 7
S6:    andi  t5, t3, t2
S7:    sw    t4, 100(t0)
S8:    sw    t5, 104(t0)
S9:    sw    t3, 108(t0)
```

1. Bestimmen Sie alle echten Datenabhängigkeiten im Programmstück. 4 P.

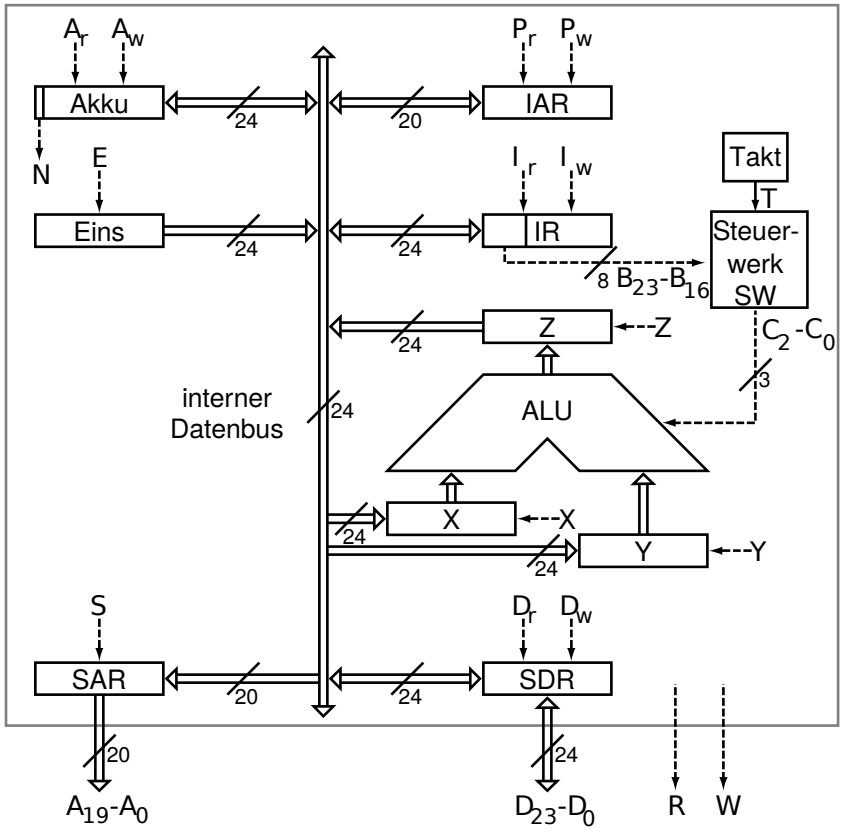
2. Nehmen Sie an, dass keine *Forwarding* Techniken implementiert sind und die auftretenden Pipelinekonflikte durch Einfügen von NOP (*No Operation*) Befehlen behoben werden müssen. 3 P.

   Ergänzen Sie das obige Programm, so dass es korrekte Ergebnisse liefert. Sie dürfen dabei die Reihenfolge der Befehle **nicht** ändern und nur so wenig NOP-Befehle wie möglich einfügen.

Aufgabe 11 *Verschiedenes* (5 Punkte)

1. Welchen Faktor zur Berechnung der CPU-Zeit versucht man durch RISC Befehlssatzarchitekturen zu optimieren? 1 P.
2. Nennen Sie die sechs grundlegenden Komponenten, die zur Funktion eines allgemeinen Schnittstellenbausteins notwendig sind. 2 P.
3. Welche zwei Hauptaufgaben übernimmt die Arbitrierung bei einem Bus? 1 P.
4. Was ist der Hauptunterschied zwischen den Verbindungsstrukturen PCI und PCI-Express? 1 P.

Architektur der MIMA



Register

Akku: Akkumulator
X: 1. ALU Operand
Y: 2. ALU Operand
Z: ALU Ergebnis
Eins: Konstante 1
IAR: Instruktionsadressregister
IR: Instruktionsregister
SAR: Speicheradressregister
SDR: Speicherdatenregister

Steuersignale vom SW

– für den internen Datenbus

A_r: Akku liest
A_w: Akku schreibt
X: X-Register liest
Y: Y-Register liest
Z: Z-Register schreibt
E: Eins-Register schreibt
P_r: IAR liest
P_w: IAR schreibt
I_r: IR liest
I_w: IR schreibt
D_r: SDR liest
D_w: SDR schreibt
S: SAR liest

– für die ALU

C_2-C_0: Operation auswählen

– für den Speicher

R: Leseanforderung
W: Schreibanforderung

Meldesignale zum SW

T: Takteingang
N: Vorzeichen des Akku
B_{23}-B_{16}: OpCode-Feld im IR

$C_2C_1C_0$	*ALU Operation*
0 0 0	tue nichts (d.h. Z → Z)
0 0 1	X + Y → Z
0 1 0	rotiere X nach rechts → Z
0 1 1	X AND Y → Z
1 0 0	X OR Y → Z
1 0 1	X XOR Y → Z
1 1 0	Eins-Komplement von X → Z
1 1 1	falls X = Y, -1 → Z, sonst 0 → Z

OpCode	*Mnemonik*	*Beschreibung*
0	LDC c	C → Akku
1	LDV a	<a> → Akku
2	STV a	Akku → <a>
3	ADD a	Akku + <a> → Akku
4	AND a	Akku AND <a> → Akku
5	OR a	Akku OR <a> → Akku
6	XOR a	Akku XOR <a> → Akku
7	EQL a	falls Akku = <a>:-1 → Akku sonst : 0 → Akku
8	JMP a	a → IAR
9	JMN a	falls Akku < 0 : a → IAR
F0	HALT	stoppt die MIMA
F1	NOT	bilde Eins-Komplement von Akku → Akku
F2	RAR	rotiere Akku eins nach rechts → Akku

Befehlsformate