Rechnerarchitektur

Einführung in die Informatik & Rechnerarchitektur (EIR1/EIF1)

Erik Pitzer

SE & MBI – FH Hagenberg – WS 2025/26

Mikroprogrammierung

Ablaufdiagramm

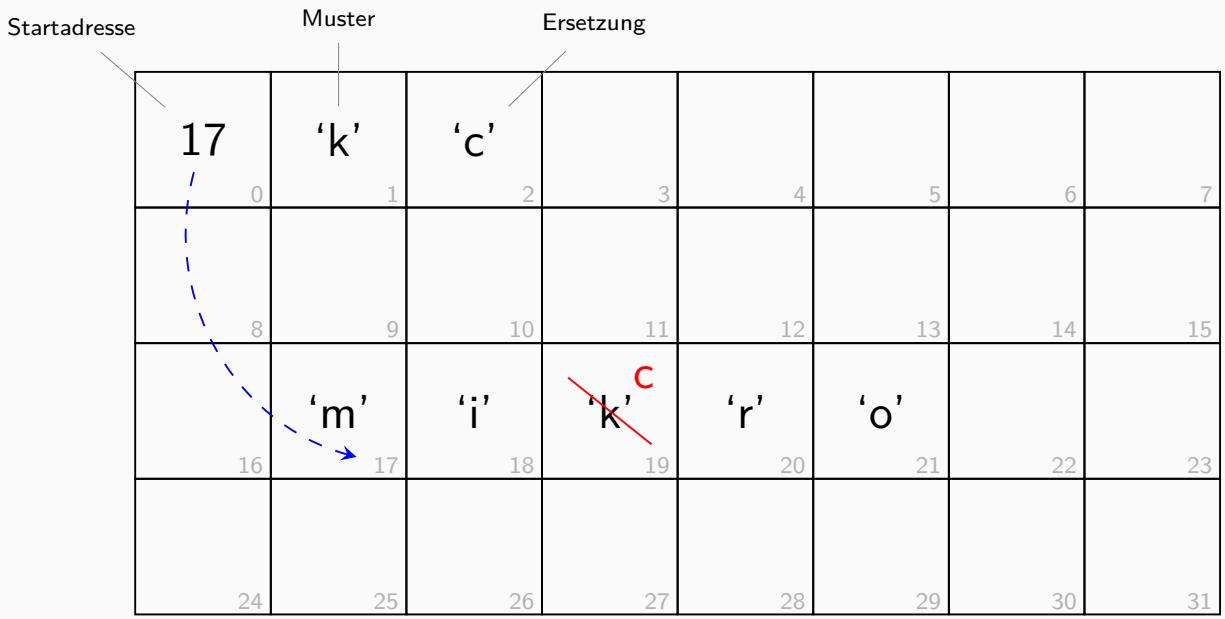
- kann mehrere Mikrobefehle enthalten, im Beispiel 2 und 3
- kann den Ablauf ändern
  - kann entweder
    - Springen: Bedingung erfüllt (`Ja` oder auch `Nein`)
    - oder zum nächsten Befehlt weiter laufen (`sonst`)
  - braucht Vorbereitung
    - Segmentnummer muss vorher in COP Register gebracht werden

Erweitern des Ablaufdiagramms

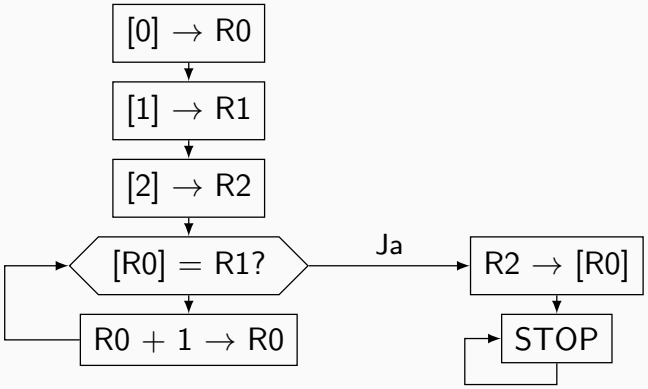
1. Sprungvorbereitung einplanen
   - jede Verzweigung (und jeder berechnete Sprung) benötigt die Zielsegmentnummer im COP-Register
   - muss über Umweg geladen werden (Z $\rightarrow$ MDR, MDR $\rightarrow$ COP)
2. Sprungziele identifizieren und eventuell anpassen
   - jedes Sprungziel muss am Start eines Segmentes liegen (an einer durch vier teilbaren Adresse)
   - Falls Sprung direkt nach SVB, muss Sprung eventuell angepasst werden
3. Mikrobefehle "zählen"
   - Speicherzugriffe [x] und Sprungvorbereitung (SVB) $\rightarrow$ 2 Takte
   - alle anderen Operationen $\rightarrow$ 1 Takt
4. Segmentnummern für Sprungziele eintragen
   - wird später als Zielwert für COP oder CN benötigt

Search and Replace

- Beginne Suche ab der Adresse die in Speicherzelle `[0]` steht
- Suche gleichen Wert, wie in Speicherzelle `[1]`
- Ersetze gefundenen Wert mit dem Wert der in Zelle `[2]` steht



Search and Replace Ablaufdiagramm

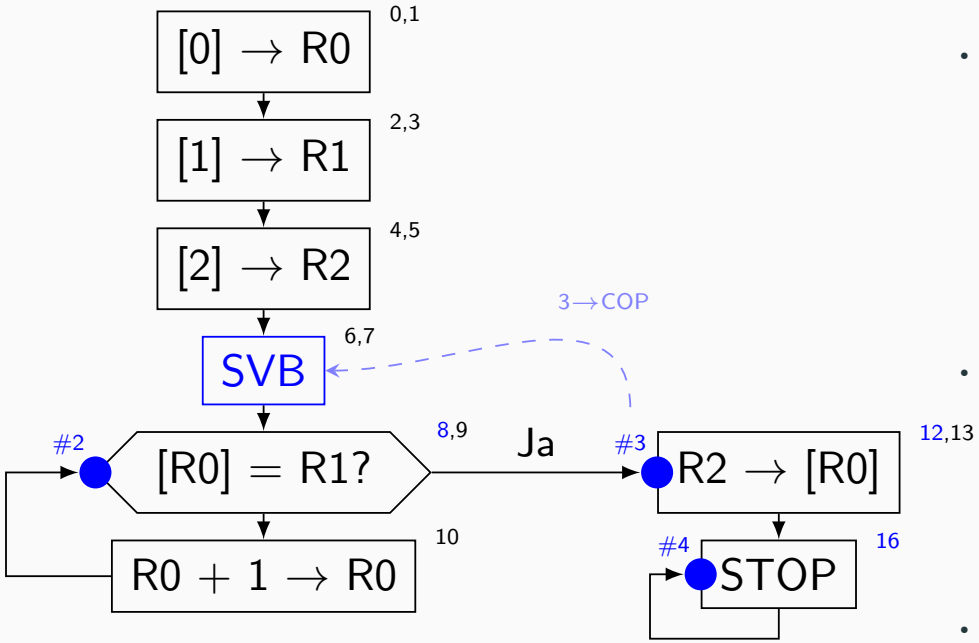


Zuerst werden die “Parameter” aus dem Speicher in Register geladen.

- in `R0` ist der aktive Index und wird mit dem Wert aus Speicherzelle `0` initialisiert.
- in `R1` ist das zu suchende Zeichen bzw. der ASCII-Code davon.
- in `R2` ist das Zeichen, dass stattdessen geschrieben werden soll.

- Dann wird der Wert im Speicher an der Adresse `R0`, also am aktiven Index, gelesen und mit dem zu suchenden Zeichen in `R1` vergleichen.
- Falls “Ja”, also beide Werte gleich sind, wird die Ersetzung in den Speicher am aktuellen Index geschrieben.
- Falls nicht, wird der Index erhöht und weiter gesucht, also zurück gesprungen zum Vergleich.

Search and Replace Erweitertes Ablaufdiagramm



- Hier wurde die Sprungvorbereitung direkt vor der Verzweigung eingeplant, dort kann **COP** auf die richtige Adresse gesetzt werden.
- Die Sprungziele wurden markiert und es wurde darauf geachtet, dass alle Schritte zu denen gesprungen werden muss an einer Mikrocode-Adresse liegen, die durch vier teilbar ist.
- Alle Speicherzugriffe wurden mit zwei Mikroinstruktionen veranschlagt, da zuerst immer die Adresse gesetzt werden muss.
- Die Sprungvorbereitung braucht ebenfalls zwei Instruktionen, da das **COP** Register nur über den Umweg durch das **MDR** gesetzt werden kann.

Search and Replace Mikrocode – 1. Parameter lesen

$\boxed{[0] \rightarrow R0}^{0,1}$

#0/0	H	
	R	Z := 0, CAR++
	B	Z → MAR
	RAM	warten
1	RAM	lesen ([MAR] → MDR)
	H	MDR → Y
	R	Z := Y, CAR++
	B	Z → R0

Der Startindex wird von der Adresse 0 aus dem Speicher gelesen und ins Register gebracht.

- Dazu muss zuerst die Adresse als Konstante generiert werden. In der ALU wird also die Zahl Null erzeugt (`Z:=0`) und in das Adressregister `MAR` gebracht (`Z` → `MAR`).
- Dann kann der Wert aus dem Speicher gelesen werden (`lesen`) und landet im `MDR`. Von dort wird der Wert zur ALU geholt (`MDR` → `Y`), durch die ALU geschleust (`Z:=Y`) und dann nach `R0` gebracht.

Search and Replace Mikrocode – 2. & 3. Parameter lesen

`[1] → R1` [2,3]

```
2   R     Z := 1, CAR++
    B     Z → MAR
    RAM   warten
-------------------------------
3   RAM   lesen ([MAR] → MDR)
    H     MDR → Y
    R     Z := Y, CAR++
    B     Z → R1
```

```
#1/4   R     Z := 2, CAR++
       B     Z → MAR
       RAM   warten
-------------------------------
  5    RAM   lesen ([MAR] → MDR)
       H     MDR → Y
       R     Z := Y, CAR++
       B     Z → R2
```

Die nächsten beiden Schritte funktionieren nach “Kochrezept” genau wie der erste Schritt.

- Es müssen nur die Adressen und statt verwenden werden
- und die Werte zu den Registern und statt gebracht werden.

Search and Replace Mikrocode – Sprungvorbereitung

SVB [6,7]

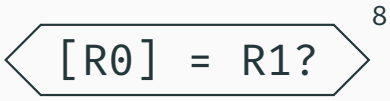
```
Vorbereitung Sprung zu Segment #3

6   H
    R       Z := 3, CAR++
    B       Z → MDR
    RAM     warten
-------------------------
7   H       MDR → COP
    R       Z := Z, CAR++
    B
    RAM     warten
```

Da im übernächsten Schritt ein bedingter Sprung zum Segment 3 gemacht wird, muss dieser zuerst vorbereitet werden. Der bedinge Sprung nimmt die Segmentnummer des Sprungziels aus dem COP Register. Dieses ist aber nur über MDR erreichbar.

- Daher wird im ersten Schritt die Segmentnummer 3 von der ALU generiert (Z:=3) und dann in das MDR gebracht (Z→ MDR).
- Im nächsten Schritt wird der Wert von dort ins COP gebracht (MDR→ COP). Ansonsten ist nicht zu tun (Z:=Z).

Search and Replace Mikrocode – Vergleich




```
#2/8   H      R0 → X
       R      Z := X, CAR++
       B      Z → MAR
       RAM    warten
-------------------------------
    9  RAM    lesen
       H      MDR → Y, R1 → X
       R      Z := X - Y
       AR     CAR=4·COP IF cond≠0
       AR     MASKE = 1000 (Z+-Of)
```

Hier soll der Wert im Register R1 mit dem nächsten Zeichen im Speicher verglichen werden. Dieses Zeichen steht im Speicher an der Stelle deren Adresse in R0 liegt. Falls beide Werte gleich sind, wird der vorher vorbereitete Sprung ausgeführt, sonst wird der nächste Befehl ausgeführt (CAR++). Da dieser Befehl ein Sprungziel ist, muss seine Adresse (8) durch vier teilbar sein.

- Zuerst muss also R0 als Adresse gesetzt werden, damit dann der Wert an dieser Adresse aus dem Speicher geholt werden kann ([R0]). Dazu wird R0 über X durch die ALU geschleust (Z:=X) und dann in das MAR gebracht.
- Nun kann gelesen werden (lesen) und der Wert über das MDR nach Y geholt werden (MDR→ Y). Gleichzeitig wird der Wert aus R1 zum Vergleich geholt (R1→ X). Ein Vergleich lässt sich durch eine Subtraktion realisieren, dabei wird das Ergebnis genau dann Null wenn beide Werte gleich sind. Das Zero-Flag (Z) kann nach der Subtraktion als Entscheidung, ob ein Sprung gemacht werden soll, herangezogen werden. Der Sprungmodus wird eingestellt (CAR=4COP IF cond<>0) und die MASKE entsprechend gesetzt.

Search and Replace Mikrocode – Inkrement & Loop

`R0 + 1 → R0` [10]

und Rücksprung zu Segment #2

10	H	R0 → X
	R	Z := INC X
	AR	CAR=4·CN, CN=2
	B	Z → R0
	RAM	warten

Falls im letzten Schritt nicht gesprungen wurde wird zuerst der Index (R0) erhöht und dann gleich wieder zurück zum Vergleich am Anfang des Segments gesprungen.

Dies ist ein fixer Sprung, er hängt also nicht von einer Berechnung ab. Daher kann die Nummer des Zielsegments direkt im Mikrobefehl stehen und muss nicht erst in das COP Register gebracht werden.

- R0 wird zur ALU geholt (R0→ X), dort inkrementiert (Z:=INC X) und wieder zurückgeschrieben (Z→ R0).
- Gleichzeitig wird der Sprungmodus so konfiguriert, dass der nächste Mikrobefehl von der Adresse 8 geholt wird (4· CN, CN=2), und damit ein Rücksprung zum Vergleich am Anfang des Segments.

Search and Replace Mikrocode – Schreiben

```
R2 → [R0]   12,13
und fixer Sprung zu
Segment #4
```

```
#3/12   H     R0 → X
        R     Z := X, CAR++
        B     Z → MAR
        RAM   warten
-------------------------------------
   13   H     R2 → X
        R     Z := X
        AR    CAR=4·CN, CN = 4
        B     Z → MDR
        RAM   schreiben (MDR→[MAR])
```

In diesem Schritt soll der Wert von R2 in den Hauptspeicher geschrieben werden, und zwar an die Adresse, die in R0 liegt ([R0]). Auch hierher muss wieder gesprungen werden, deswegen liegt der erste Befehl auf Adresse 12 im Segment #3.

- Dazu wird zuerst wieder die richtige Adresse eingestellt, also R0 geholt (R0→X) durchgeschleust (Z:=X) und dann in das Adressregister MAR geschrieben (Z→MAR).
- Dann wird der Wert aus R2 geholt (R2→X), ebenfalls durchgeschleust (Z:=X) und ins Datenregister MDR gebracht (Z→MDR). Dann muss nur noch der Speichermodus auf **schreiben** gesetzt werden damit der Wert aus MDR in den Speicher geschrieben wird.
- Gleichzeitig wird noch ein fixer Sprung zum Schritt STOP eingestellt (CAR=4CN, CN=4).

Search and Replace Mikrocode – Stop-“Schleife”

```
STOP 16   Endlosschleife Sprung
          zu aktuellem Segment #4

#4/16     H
          R    CAR = 4·CN, CN=4
          B
```

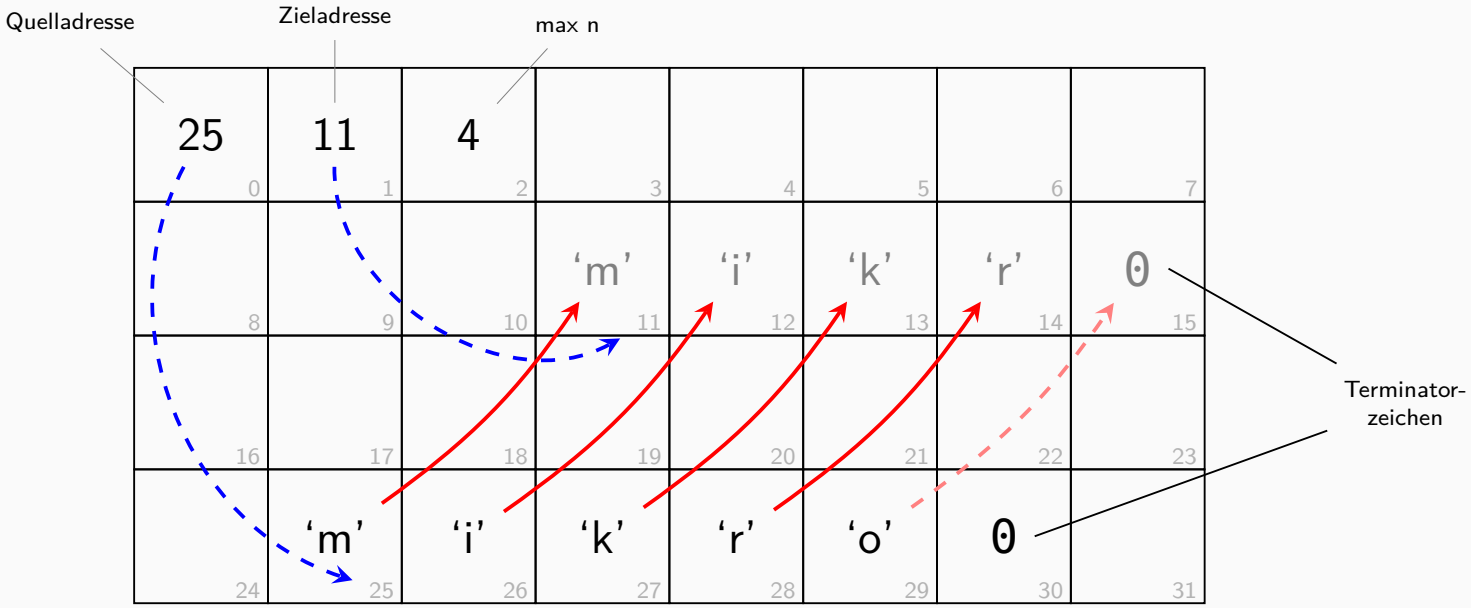
Hier befindet sich eine Endlosschleife. Ein Befehl, der immer wieder zu sich selbst springt. Daher muss auch dieser Befehl an einer durch vier teilbaren Adresse (16) liegen.

Hier wird einfach immer wieder zum aktuellen Mikrobefehl gesprungen. Dadurch beliebt der Simulator scheinbar stehen, auch wenn natürlich der Takt und die CPU weiterlaufen.

- Es muss nur der Sprungmodus auf fixen Sprung gestellt werden (CAR=4CN) und der Wert CN direkt im Mikrobefehl auf die richtige Segmentnummer (4) gesetzt werden.

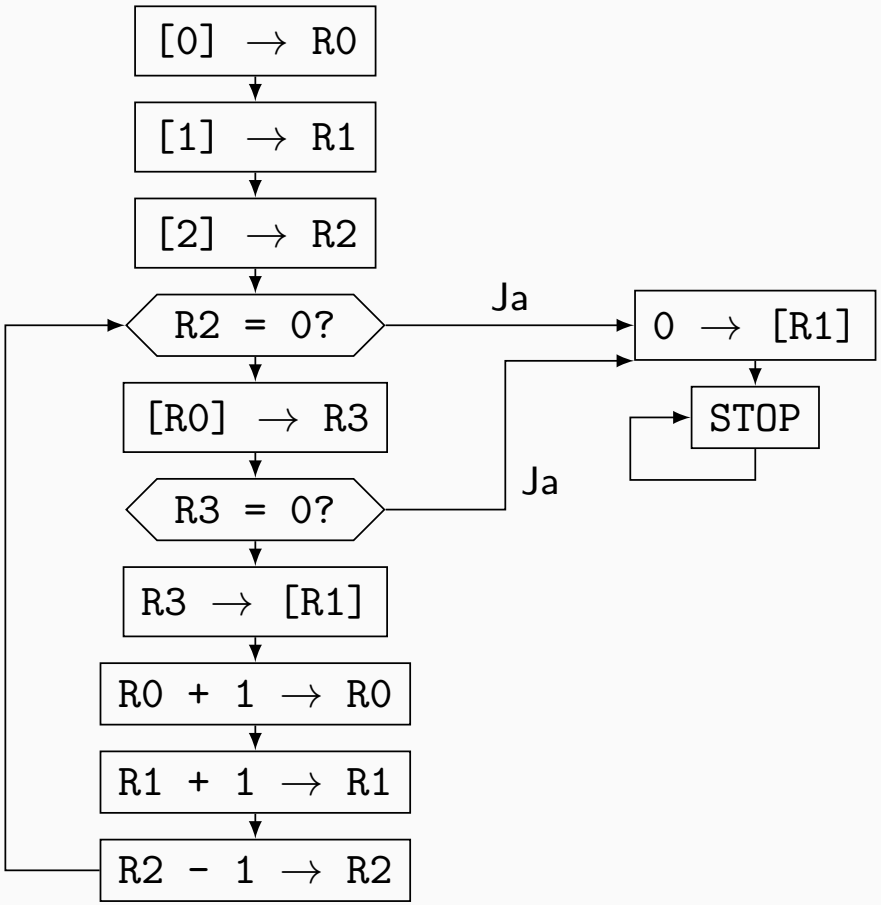
String Copy – Aufgabenstellung

- Kopiere Zeichenkette ab Adresse die in Speicherzelle `[0]` steht
- an Adresse die in Speicherzelle `[1]` steht.
- Brich ab sobald
  - das Terminatorzeichen (ASCII-Code Null) kopiert wurde,
  - oder *n* Zeichen kopiert wurden, *n* steht im Speicher an der Stelle `[2]`



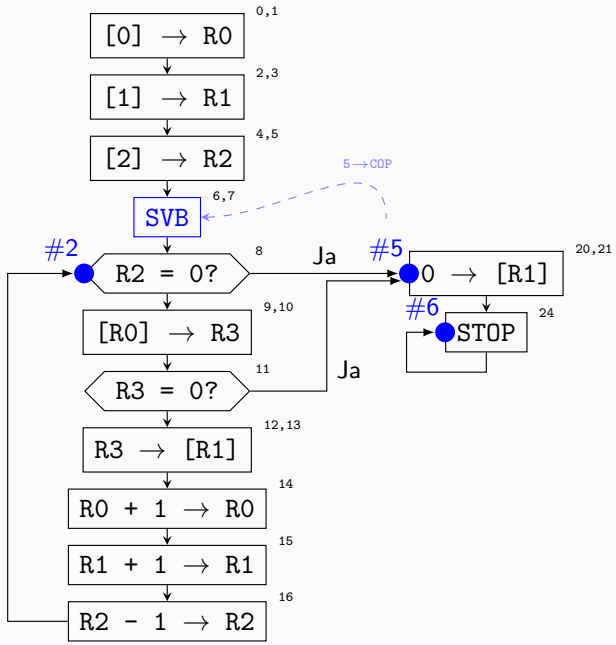
String Copy – Ablaufdiagramm

- R0 … Quellindex
- R1 … Zielindex
- R2 … Anzahl (max) restlicher Zeichen
- R3 … Puffer



- Zuerst werden Parameter aus dem Speicher `[0-2]` in Register `R0-R2` gelesen.
- Falls die Anzahl zu kopierender Zeichen `0` ist, wird abgebrochen (`R2=0?`).
- Dann wird der Wert im Speicher an der Adresse die in `R0` steht nach `R3` gelesen (`[R0]`→ `R3`).
- Falls dieses Zeichen `0` ist wird abgebrochen (`R3=0?`).
- Falls nicht wird das gelesene Zeichen in `R3` in den Speicher an Adresse `R1` geschrieben (`R3\to [R1]`)
- Zuletzt werden Lese- und Schreibindex (R0 und R1) erhöht und die Anzahl restlicher Zeichen verringert (`R2`), dann Rücksprung zum Anfang.

String Copy – Erweitertes Ablaufdiagramm



Vor jeder Verzweigung muss eine Sprungvorbereitung kommen (Zielsegment → COP), hier sind beide bedingten Sprünge zu gleichem Ziel, daher genügt eine gemeinsame Sprungvorbereitung direkt vor der ersten Verzweigung

- Aus dem gleichen Grund genügt es auch, beim Rücksprung unter die Sprungvorbereitung zu springen, da sich der Wert in COP nicht ändern muss.
- Als Sprungziele werden dann der erste Vergleich (`R2=0?`), das finale Schreiben des Terminatorzeichens (`0→ [R1]`) und die Stop-“Schleife” markiert
- Für alle Instruktionen mit Speicherzugriff sowie für die Sprungvorbereitung werden zwei Instruktionen eingeplant, sonst eine.
- Die Sprungziele müssen an einer durch vier teilbaren ROM-Adresse liegen.
- Bei Befehl 8 (`R2=0?`) ist das zufällig ohnehin der Fall, Befehl 22 (`0→ [R1]`) muss verschoben werden.