



SR/SRU  
SL/SLU

## Arithmetisch/logische Befehle

Shift

Logischer / arithmetischer Shift von \$Y um \$Z / Z Stellen

Linksshift : **SLU / SL \$X,\$Y,\$Z / Z**

Zielregister    Quellregister    Shiftzahl



Rechtsshift : **SRU / SR \$X,\$Y,\$Z / Z**





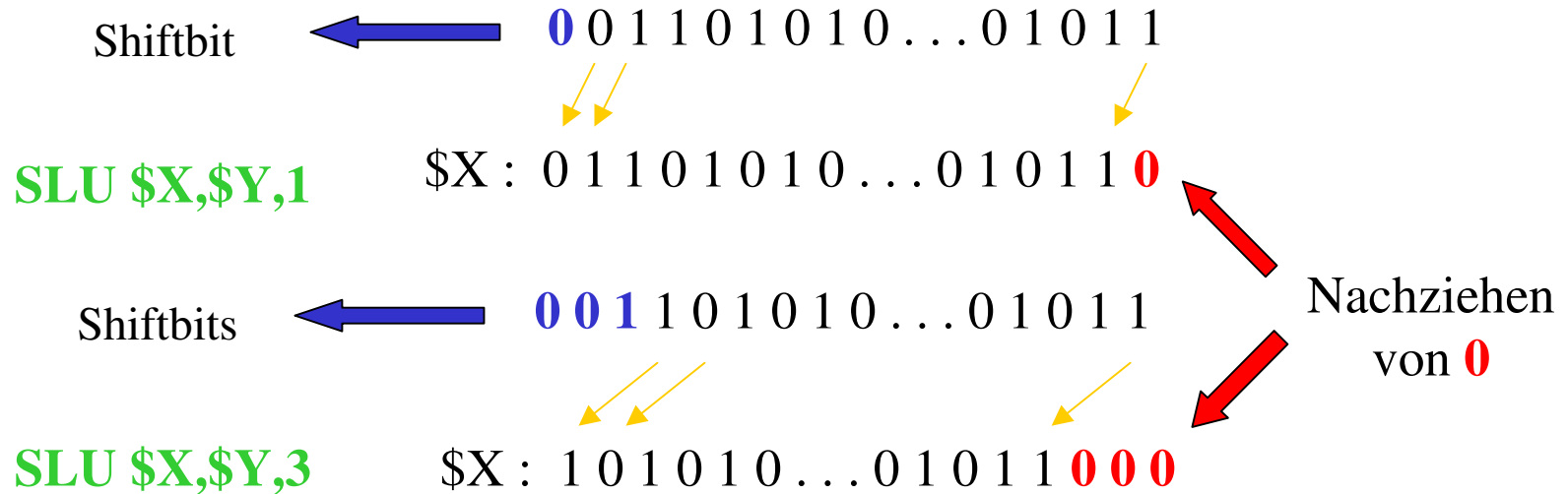
SR/SRU  
SL/SLU

## Arithmetisch/logische Befehle

Shift

**SLU \$X,\$Y,\$Z/Z**

Logischer Shift von \$Y um \$Z / Z Stellen nach links



⚡ Die Shiftbits gehen verloren



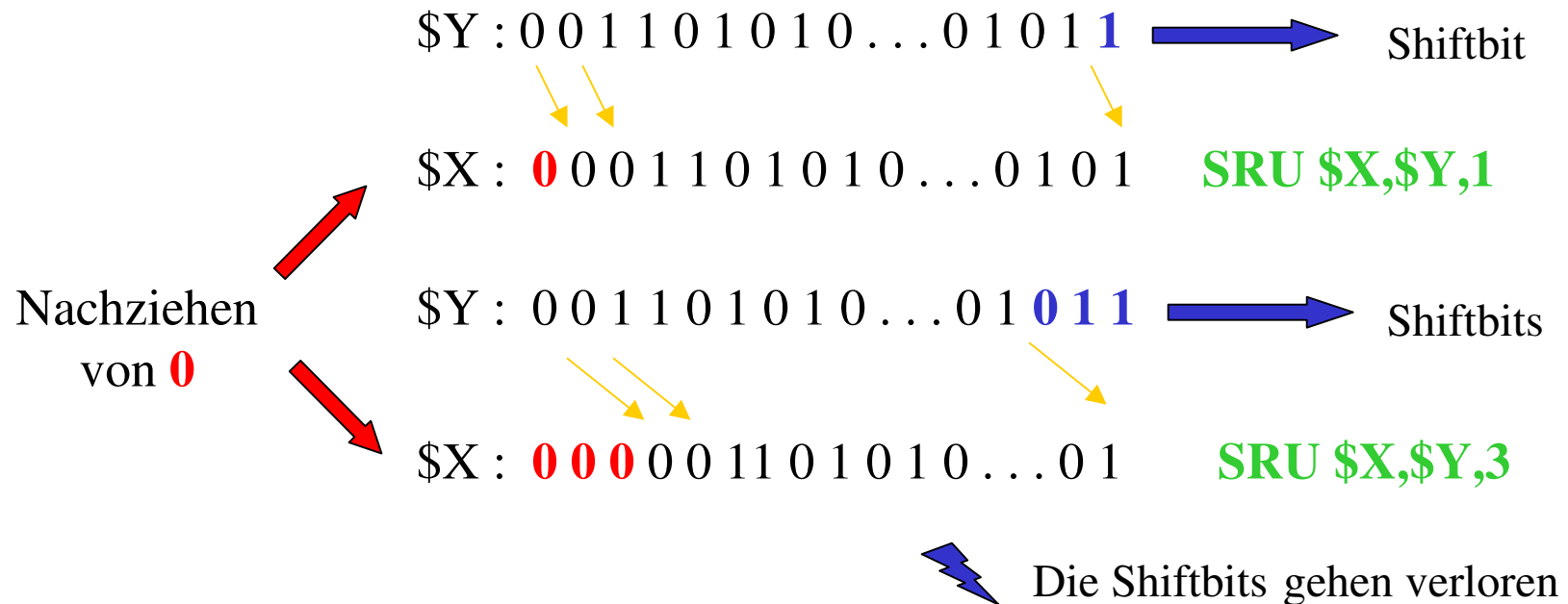
SR/SRU  
SL/SLU

### Arithmetisch/logische Befehle

Shift

**SRU \$X,\$Y,\$Z/Z**

Logischer Shift von \$Y um \$Z / Z Stellen nach rechts





SR/SRU  
SL/SLU

## Arithmetisch/logische Befehle

Logischer Shift

### Beispiele

```
SETH $1,#1234  
SRU  $2,$1,4  
SLU  $3,$1,4
```

...

mmix>

e0011234 (SETH) \$1 = #1234 0000 0000 0000

3f020104 (SRUI) rL=3,

\$2 = #1234 0000 0000 0000 >>4  
= #0123 4000 0000 0000

3b030104 (SLUI) rL=4,

\$3 = #1234 0000 0000 0000 <<4  
= #2340 0000 0000 0000



SR/SRU  
SL/SLU

## Arithmetisch/logische Befehle

Linksshift/Rechtsshift

### Beispiele

...

```
SETH $1,#ffff  
SRU  $2,$1,4  
SLU  $3,$1,4
```

`mmix>`

```
e001ffff (SETH) $1 = #ffff 0000 0000 0000
```

```
3f020104 (SRUI)
```

```
$2 = #ffff 0000 0000 0000 >> 4
```

```
= #0fff f000 0000 0000
```

```
3b030104 (SLUI)
```

```
$3 = #ffff 0000 0000 0000 << 4
```

```
= #fff0 0000 0000 0000
```

rA = ?



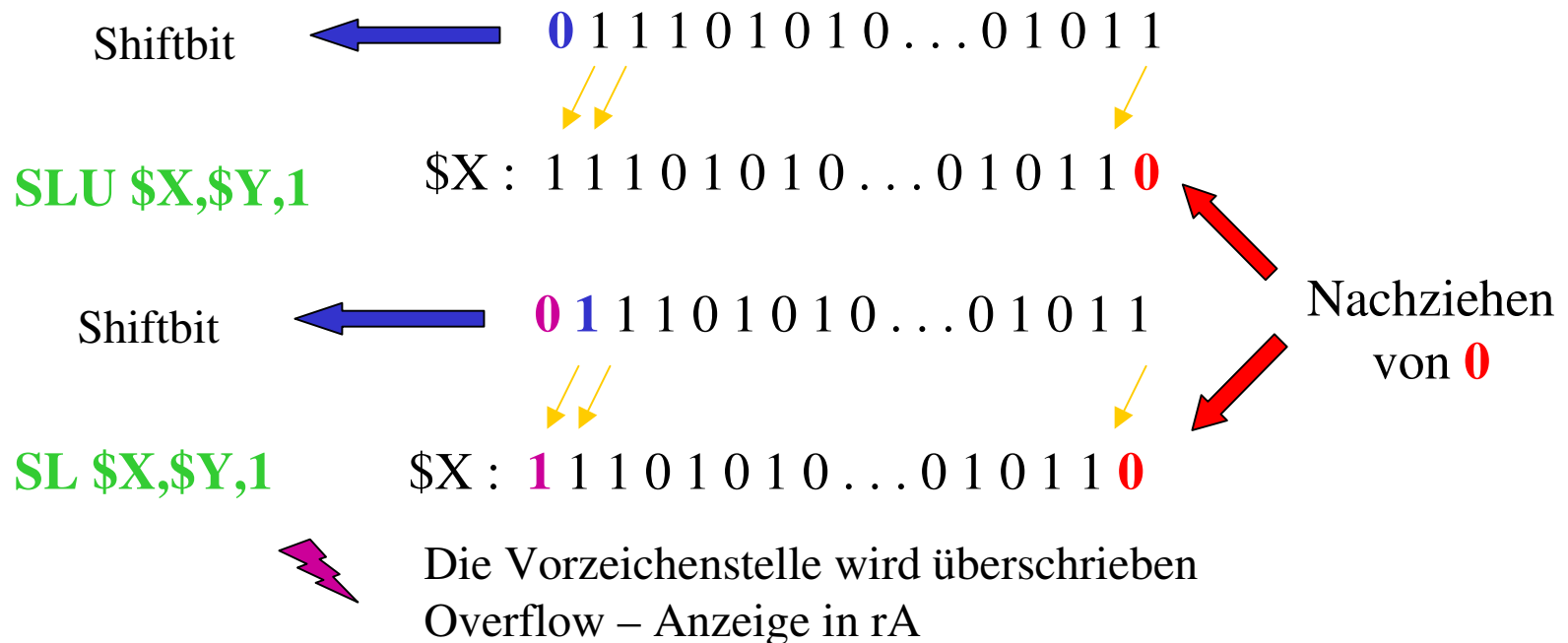
SR/SRU  
SL/SLU

## Arithmetisch/logische Befehle

Shift

**SL \$X,\$Y,\$Z/Z**

Arithmetischer Shift von \$Y um \$Z / Z Stellen nach links





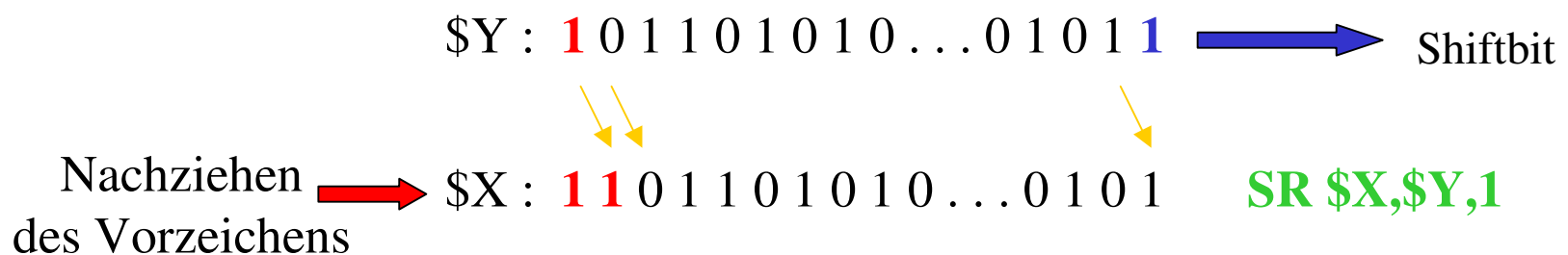
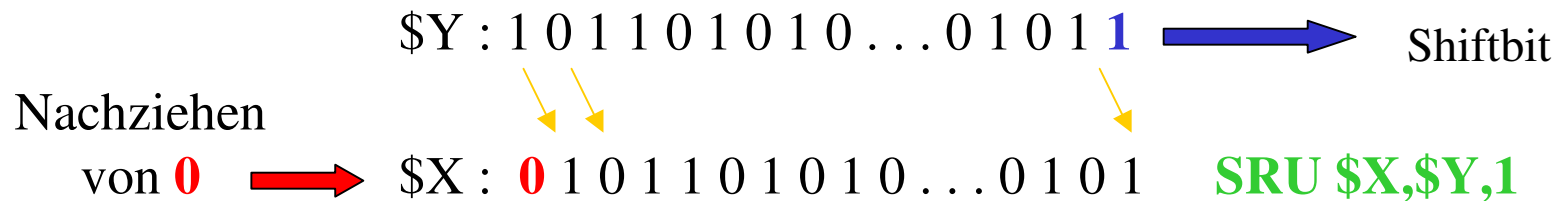
SR/SRU  
SL/SLU

## Arithmetisch/logische Befehle

Shift

**SR \$X,\$Y,\$Z/Z**

Arithmetischer Shift von \$Y um \$Z / Z Stellen nach rechts





SR/SRU  
SL/SLU

## Arithmetisch/logische Befehle

Linksshift/Rechtsshift

### Beispiele

SETH \$1,#1234  
SR \$2,\$1,4  
SL \$3,\$1,4

...

Overflow! 

`mmix>`

`e0011234 (SETH) $1 = #1234 0000 0000 0000`

`3d020104 (SRI) rL=3,`

`$2 = 1311673391471656960 >> 4 =`

`81979586966978560 = #0123 4000 0000 0000`

`39030104 (SLI) rL=4,`

`$3 = 1311673391471656960 << 4 =`

`2540030189836959744 = #2340 0000 0000 0000`

`rA = #00040`



SR/SRU  
SL/SLU

## Arithmetisch/logische Befehle

Linksshift/Rechtsshift

### Beispiele

`mmix>`

...

SETH \$1,#ffff  
SR \$2,\$1,4  
SL \$3,\$1,4

`e001ffff (SETH) $1 = #ffff 0000 0000 0000`

`3d020104 (SRI)`

`$2 = -281474976710656 >> 4 =`

`-17592186044416 = #ffff f000 0000 0000`

`39030104 (SLI)`

`$3 = -281474976710656 >> 4 =`

`-4503599627370496 = #fff0 0000 0000 0000`

`rA = ?`



SR/SRU  
SL/SLU

## Arithmetisch/logische Befehle

Linksshift/Rechtsshift

### Beispiele

...  
SETH \$1,#7fff  
SLU \$2,\$1,1  
SL \$3,\$1,1

mmix>

e0017fff (SETH)

\$1 = #7fff 0000 0000 0000

3b020101 (SLUI)

\$2 = #7fff 0000 0000 0000 << 1

= #fffe 0000 0000 0000

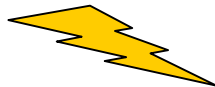
39030101 (SLI)

\$3 = 9223090561878065152 << 1

= -562949953421312,

rA=#00040

Overflow !





SR/SRU  
SL/SLU

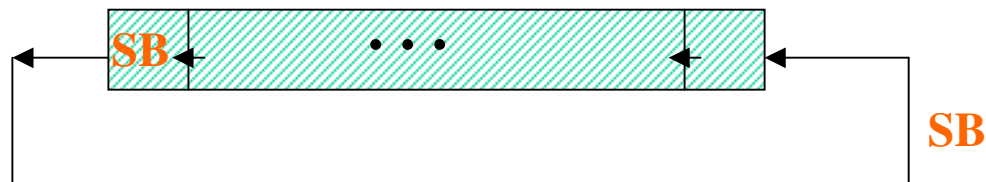
Arithmetisch/logische Befehle

Linksshift/Rechtsshift

## Übung

logischer Kreisshift links

Das links herausgefallene Bit wird rechts wieder eingespeist





SR/SRU  
SL/SLU

## Arithmetisch/logische Befehle

Linksshift/Rechtsshift

### Lösungsvorschlag

- \* SHIFT-Befehle
- \* Kreisshift links

LOC #100

Main	SETH	\$1,#aaaa	Testmuster 1010 ...
	SLU	\$2,\$1,1	
	SRU	\$3,\$2,1	
	XOR	\$3,\$1,\$3	Shiftbit links
	SRU	\$3,\$3,63	Shiftbit rechts
	OR	\$2,\$2,\$3	Ergebnis in \$2
	TRAP	0,Halt,0	



SR/SRU  
SL/SLU

## Arithmetisch/logische Befehle

Linksshift/Rechtsshift

### Ablauf

mmix>

(SETH) \$1=1[1] = #aaaa 0000 0000 0000

(SLUI) \$2=1[2] = #aaaa 0000 0000 0000 << 1 = #5554 0000 0000 0000

(SRUI) \$3=1[3] = #5554 0000 0000 0000 >> 1 = #2aaa 0000 0000 0000

(XOR) \$3=1[3] = #aaaa 0000 0000 0000  
          ^ #2aaa 0000 0000 0000 = #8000 0000 0000 0000

(SRUI) \$3=1[3] = #8000 0000 0000 0000 >> 63 = #0000 0000 0000 000**1**

(OR) \$2=1[2] = #5554 0000 0000 0000  
      | #0000 0000 0000 000**1** = #5554 0000 0000 000**1**



SR/SRU  
SL/SLU

## Arithmetisch/logische Befehle

Linksshift/Rechtsshift

### Alternative

- \* SHIFT-Befehle
- \* Kreisshift links

LOC #100

Main	SETH	\$1,#aaaa	Testmuster 1010 ...
	SRU	\$2,\$1,63	Shiftbit rechts
	SLU	\$3,\$1,1	
	OR	\$2,\$2,\$3	Ergebnis in \$2
	TRAP	0,Halt,0	



SR/SRU  
SL/SLU

## Arithmetisch/logische Befehle

Linksshift/Rechtsshift

### Multiplikation durch Linksshift

Beispiel:

SL ...1	0...0001 = 1	0...0011 = 3
SL ...1	0...0010 = 2	0...0110 = 6
	0...0100 = 4	0...1100 = 12
	...	...

Linksshift um n Stellen bewirkt eine Multiplikation mit  $2^n$



Overflow – Anzeige in rA



SR/SRU  
SL/SLU

## Arithmetisch/logische Befehle

Linksshift/Rechtsshift

### Division durch Rechtsshift

Beispiel:

	↑	0...0001 = 1	0...0011 = 3
SR ...1		0...0010 = 2	0...0110 = 6
SR ...1		0...0100 = 4	0...1100 = 12
		...	...

Rechtsshift um  $n$  Stellen bewirkt eine Division durch  $2^n$



Kein Rest in rR



SR/SRU  
SL/SLU

## Arithmetisch/logische Befehle

Linksshift/Rechtsshift

### Multiplikation durch Linksshift

Beispiel (negative Zahlen):

SL ...1	1...1111 = -1	1...1101 = - 3
SL ...1	1...1110 = -2	1...1010 = - 6
	1...1100 = -4	1...0100 = -12
	...	...





SR/SRU  
SL/SLU

## Arithmetisch/logische Befehle

Linksshift/Rechtsshift

### Division durch Rechtsshift

Beispiel (negative Zahlen):

	↑	1...1111 = -1	1...1101 = - 3	
SR ...1		1...1110 = -2	1...1010 = - 6	✓
SR ...1		1...1100 = -4	1...0100 = -12	
		...	...	

**Aber :** 1...1101 = - 3  
 SR ...1  
 1...1110 = - 2 !!!



Bei ungeraden Zahlen  
Ergebnis um 1 zu groß



SR/SRU  
SL/SLU

## Arithmetisch/logische Befehle

Linksshift/Rechtsshift

### Division durch Rechtsshift

Beispiel (negative Zahlen):

Verwende Vorzeichen-Betrag-Darstellung

$$\begin{array}{r} -3 = \mathbf{1} 00 \dots 011 \\ \text{Vorzeichen} \downarrow \underbrace{\hspace{10em}} \text{SR ...1} \\ \mathbf{1} 00 \dots 001 = -1 \quad \checkmark \end{array}$$

Keine Anzeige des Rests



SR/SRU  
SL/SLU

Arithmetisch/logische Befehle

Linksshift/Rechtsshift

### Ausführungszeiten

MUL 10

DIV 30

SL/SR 1 oops





SR/SRU  
SL/SLU

Arithmetisch/logische Befehle

Linksshift/Rechtsshift

## Übung

Division durch  $2^n$  mit Bestimmung des Rests

