Tobias Eidelpes
2015WS

Technische Grundlagen der Informatik Übungsblatt 5

Aufgabe 2.

Lösung.

(1)	Bei der Micro16 Architektur sind Datenwörter 8 Bit lang	falsch
(2)	Bei der Micro16 Architektur werden Mikroinstruktionen durch eine 16 Bit Adresse adressiert	falsch
(3)	Das *Memory Address Register* (MAR) ist mit dem Adressbus verbunden.	richtig
(4)	Liegt an der *Memory Select* (MS) Leitung logisch 0 an, wird in das *Memory Buffer Register* (MBR) geschrieben, ansonsten wird geladen.	falsch
(5)	Micro-Instruktionen sind beim Micro16 genau 16 Bit lang.	falsch
(6)	Die *Micro Sequencing Logic* kann Sprünge ausführen, indem sie den Folgewert des *Micro Instruction Counters* (MIC) bestimmt.	richtig
(7)	Am Ausgang Z der ALU liegt genau dann logisch 1 an, wenn im Ergebnis alle Bit logisch 0 sind.	richtig
(8)	Mithilfe des S-Bus Decoders werden die Registerwerte über den A-Bus transferiert	falsch
(9)	Die ALU-Operation Negation invertiert ein Datenwort bitweise.	richtig
(10)	Die Operation *right shift* entspricht einer Multiplikation des Datenworts mit der Zahl 2.	falsch
(11)	Die Control Unit teilt den Takt in drei Phasen und steuert damit die Ausführung der Mikroinstruktionen.	falsch

Aufgabe 3.

Lösung.

(1)	❍	`R9 ← R9∨R8`
(2)	☒	`R8 ← lsh(R1∧R2)`
(3)	☒	`R5 ← R6+R12`
(4)	❍	`R9 ← 2`
(5)	❍	`R0 ← R1-R2`
(6)	☒	`rsh(1); if Z goto 5`
(7)	❍	`MAR ← MBR; rd`
(8)	❍	`R3 ← lsh((¬R2)+R3))`
(9)	☒	`goto .R4`
(10)	☒	`R1 ← (R2∧1); if Z goto nextStep`
(11)	❍	`MBR ← R1; R5 ← R0+R1`
(12)	☒	`MAR ← R1; MBR ← R0+R1; R2 ← R0+R1; wr`

Aufgabe 4.

Lösung.

(a)

	:2	
219	|	1
109	|	1
54	|	0
27	|	1
13	|	1
6	|	0
3	|	1
1	|	1

	:2	
73	|	1
36	|	0
18	|	0
9	|	1
4	|	0
2	|	0
1	|	1

$x = (219)_{10} = (1101\,1011)_2$ $y = (73)_{10} = (0100\,1001)_2$

(b)

$x = (0000\,0000\,1101\,1011)_2$ im Zweierkomplement

$y = (0000\,0000\,0100\,1001)_2$ im Zweierkomplement

(c)

```
R5 <- lsh(1)
R5 <- R5+1
R5 <- lsh(R5+R5)
R5 <- R5+1
R5 <- lsh(R5)
R5 <- R5+1
R5 <- lsh(R5+R5)
R5 <- R5+1
R5 <- lsh(R5)
R5 <- R5+1
```

```
R6 <- 1
R6 <- lsh(R6+R6)
R6 <- lsh(R6)
R6 <- lsh(R6+1)
R6 <- lsh(R6+R6)
R6 <- R6+1
```

(d)

loop	Register R5	Register R6
1	219	73
2	438	36
3	876	18
4	1752	9
5	3504	4
6	7008	2
7	14016	1

(e)

```
R7 <- 0                         #setze R7 (enthält Ergebnis) auf 0
:loop
  (R6&1); if Z goto .zero       #wenn R6 nicht 1 ist gehe zu .zero
  R7 <- R7+R5                   #addiere zu R7 den Multiplikand
:zero
  R6 <- rsh(R6)                 #shifte R6 um eine Stelle nach rechts
  (R6); if Z goto .end          #wenn R6 0 ist, gehe zu .end ansonsten
  R5 <- lsh(R5)                 #shifte R5 um eine Stelle nach links
  goto .loop
:end
```

Aufgabe 5.

Lösung.

(a)

AMUX	COND	ALU	SH	MBR	MAR	RD/WR	MS	ENS	S-BUS	B-BUS	A-BUS	ADDR
0	00	11	00	0	0	0	0	1	0110	0000	0101	00000000
0	00	01	00	0	0	0	0	1	0110	0001	0110	00000000
0	00	01	00	0	0	0	0	1	0110	0110	0100	00000000
0	00	00	00	1	1	0	1	0	0000	0111	0110	00000000
0	00	00	00	0	0	0	1	0	0000	0000	0000	00000000

(b)

```
R2 <- ~R1                    #negiert die zu subtrahierende Zahl in R1
R2 <- R2+1                   #addiert 1 und speichert in R2
R2 <- R0+R2                  #addiert R2 zu R0 (zieht R2 ab)
MBR <- R2; MAR <- R3; wr     #speichert R2 in MBR
wr                           #speichert R2 in MBR
```

(c)

Diese Mikroinstruktionen ermöglichen die Subtraktion einer Zahl in R1 von einer Zahl in R0 und speichern das Ergebnis im MBR.

Aufgabe 6.

Lösung.

```
R1 <- lsh(1+1)               #schreibe 4 in R1
R1 <- lsh(R1+R1)             #erhöhe R1 auf 8 und schiebe nach links (16)
MAR <- R1; rd                #schreibe 16 in MAR; set MS=1 und RD/WR=1
rd                           #schreibe Wert an ADDR 16 in MBR
R2 <- MBR                    #kopiere Wert an MBR in R2
R3 <- lsh(1+1)               #konstruiere 5 für mod5()
R3 <- R3+1
R3 <- ~R3                    #wandle in negative Zahl um
R3 <- R3+1                   #für Subtraktion durch Addition
:loop
  R4 <- R2                   #schreibe R2 in R4
  R2 <- R2+R3; if N goto .end  #subtrahiere R3 von R2 und schreibe in R2
  goto .loop                 #wiederhole solange, bis R2 negativ
:end
  R1 <- R4                   #schreibe Rest in R1
```

Aufgabe 7.

Lösung.

```
R10 <- 0                    #set Hamming-Distanz 0
:test                       #test if R8 == 0
  :test1
    R8; if Z goto .test2    #if true gehe zu test2
    goto .loop              #ansonsten continue main loop
  :test2                    #test if R9 == 0
    R9; if Z goto .end      #if true gehe zum Ende
    goto .loop              #ansonsten continue main loop
:loop
  R0 <- R8&1                #lsb von R8 in R0
  R1 <- R9&1                #lsb von R9 in R1
  R3 <- R0&R1               #UND-Verknüpfung der lsbs
  R3 <- ~R3                 #negiere UND-Verknüpfung
  R4 <- ~R0                 #negiere lsb von R8
  R5 <- ~R1                 #negiere lsb von R9
  R2 <- R4&R5               #UND-Verknüpfung der negierten lsbs
  R2 <- ~R2                 #negiere die UND-Verknüpfung
  R2 <- R2&R3               #verknüpfe R2 und R3 -> XOR
  R2; if Z goto .sh         #if false (lsbs gleich) goto .sh
  goto .hd                  #ansonsten erhöhe R10 um 1
:sh
  R8 <- rsh(R8)             #shifte R8 nach rechts um 1
  R9 <- rsh(R9)             #shifte R9 nach rechts um 1
  goto .test                #gehe zum Anfang
:hd
  R10 <- R10+1              #erhöhe Hamming-Distanz in R10 um 1
  goto .sh                  #shifte R8 u. R9
:end
```

$$R0 \oplus R1 = \neg(R0 \wedge R1) \wedge \neg(\neg R0 \wedge \neg R1)$$

$R0$	$R1$	$\neg$	$(R0 \wedge R1)$	$\wedge$	$\neg$	$(\neg R0 \wedge \neg R1)$
0	0	1	0	0	0	1
0	1	1	0	1	1	0
1	0	1	0	1	1	0
1	1	0	1	0	1	0

Aufgabe 8.

Lösung.

```
R0; if Z goto .end                      #if no 1s: save some cycles
R1; if Z goto .end                      #if counter == 0: save some cycles
R7 <- lsh(1+1)                          #construct MSB (-32768)
R7 <- lsh(R7+R7)
R7 <- lsh(R7+R7)
R7 <- lsh(R7+R7)
R7 <- lsh(R7+R7)
R7 <- lsh(R7+R7)
R7 <- lsh(R7+R7)
R7 <- lsh(R7)
:loop
    R2 <- (R0&1)                        #get lsb(R0)
    R3 <- ~R2                           #invert it
    R3 <- (R3&1); if Z goto .msb        #if inverted lsb == 0 gehe zu .msb
    R0 <- rsh(R0)                       #else (lsb == 0) shift right
    R1 <- R1+-1; if Z goto .end         #decrement counter; if 0: end
    goto .loop                          #repeat
:msb
    R0 <- rsh(R0)                       #if lsb == 1 shift right
    R0 <- R0+R7                         #add msb
    R1 <- R1+-1; if Z goto .end         #decrement counter; if 0: end
    goto .loop                          #repeat
:end
    R9 <- lsh(1+1)                      #construct address 16384
    R9 <- lsh(R9+R9)
    R9 <- lsh(R9+R9)
    R9 <- lsh(R9+R9)
    R9 <- lsh(R9+R9)
    R9 <- lsh(R9+R9)
    R9 <- lsh(R9+R9)
    R2 <- R9+-1                         #decrement (16383)
    MAR <- R2; MBR <- R0; wr            #copy address to MAR & save result
    wr                                  #in MBR -> save to address 16383
```