Lösungen zum 9. Übungsblatt Informatik II (SS 96)

Frank Berwanger (berwange@magicsun.cs.uni-sb.de)
Michael Mehl (mehl@dfki.uni-sb.de)

Lösung zu Aufgabe 9.1 [fib]

- fib(n) = fibiter(1,0,n)
- $\text{fibiter(a,b,n)} = \begin{cases} b & \text{falls n = 0} \\ \text{fibiter(a+b,a,n-1)} & \text{sonst} \end{cases}$

```
0     LOADI 0          ; b := 0
1     STORE 1
2     LOADI 1          ; a := 1
3     STORE 2
4     LOAD 0           ; ACC := n
5     JUMP= 10         ; if n = 0 then goto 15
6     SUBI 1           ; n := n - 1
7     STORE 0
8     LOAD 2           ; ACC := b
9     MOVE ACC IN1     ; IN1 := a
10    ADD 1            ; a := a + b;
11    STORE 2
12    MOVE IN1 ACC     ; ACC := IN1
13    STORE 1          ; b := a
14    JUMP -10         ; goto 4
```

Lösung zu Aufgabe 9.2 [Summieren]

- IN1 enthält die Adresse der zu ladenden Speicherzelle
- IN2 zählt von S(0) (>0) bis 0

1.
```
3     LOADI     0
4     STORE     2          ; S(2) := 0
5     LOAD      0
6     MOVE      ACC IN1    ; IN1:= S(0)
7     MOVE      ACC IN2    ; IN2 := S(0)
8     LOADIN1   0
9     ADD       2
10    STORE     2          ; S(2) := S(2) + S(IN1)
11    MOVE      IN1 ACC
12    ADD       1
13    MOVE      ACC IN1    ;IN1 := IN1 + 1
14    MOVE      IN2 ACC
15    SUBI      1
16    MOVE      ACC IN2    ; IN2 := IN2 - 1
17    JUMP>     -9         ; goto 8
```

2.

3	LOADI	ACC	0	
4	STORE		2	S(2) := 0
5	LOAD	IN1	0	IN1 := S(0)
6	LOAD	IN2	0	IN2 := S(0)
7	LOADIN1	ACC	0	
8	ADD	ACC	2	
9	STORE		2	S(2) := S(IN1) + S(2)
10	ADDI	IN1	1	IN1 := IN1 + 1
11	SUBI	IN2	1	IN2 := IN2 + 1
12	MOVE	IN2 ACC		
13	$\text{JUMP}_>$		-6	

Anmerkung : Programm muß eigentlich bei 0 starten!

Es gibt kein Programm mit weniger als 11 Befehlen.

Lösung zu Aufgabe 9.3 [Kodieren]

0	43000000	$LOAD\ ACC\ 0$	; lade ACC aus M(0)
1	E0000003	$JUMP_<\ 3$	; wenn $ACC < 0$, dann Ende der Schleife
2	0B000001	$SUBI\ ACC\ 1$	; Dekrementiere Akkumulator
3	$F8FFFFFE$	$JUMP\ -2$	; Wiederhole Schleife
4	80000000	$STORE\ ACC\ 0$	; Schreibe ACC nach M(0)

Lösung zu Aufgabe 9.4 [Übungsmaschine]

1. Instruktionsatz

Load Befehle	I[27:26] = D	
I[31:28]	Befehl	Wirkung
0100	LOAD D i	$D := M(\langle i \rangle)$
0101	LOADIN D	$D := M(\langle IN \rangle)$ $\quad \langle PC \rangle := \langle PC \rangle + 1$
0110	LOADI D i	$D := \langle i \rangle$

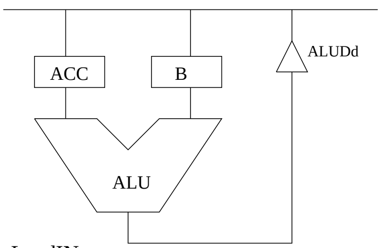
Store Befehle		
I[31:28]	Befehl	Wirkung
1000	STORE i	$M(\langle i \rangle) := ACC$ $\quad \langle PC \rangle := \langle PC \rangle + 1$
1001	STOREIN	$M(\langle IN \rangle) := ACC$

Compute Befehle		
I[31:27]	Befehl	Wirkung
00010	SUB	$ACC := [ACC] - [B]$
00011	ADD	$ACC := [ACC] + [B]$
00100	OPLUS	$ACC := ACC \oplus B$ $\quad \langle PC \rangle := \langle PC \rangle + 1$
00101	OR i	$ACC := ACC \vee B$
00110	AND i	$ACC := ACC \wedge B$

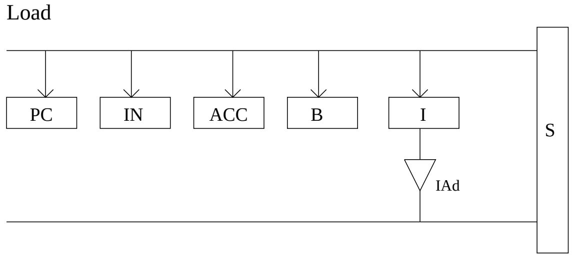
Jump Befehle		
I[31:27]	Befehl	Wirkung
11000	NOP	$\langle PC \rangle := \langle PC \rangle + 1$
11001	$\text{JUMP}_{>}$ i	
11010	$\text{JUMP}_{=}$ i	
11011	$\text{JUMP}_{\geq}$ i	$\langle PC \rangle := \begin{cases} \langle IN \rangle & \text{falls ACC c 0} \\ \langle PC \rangle + 1 & \text{sonst} \end{cases}$
11100	$\text{JUMP}_{<}$ i	
11101	$\text{JUMP}_{\neq}$ i	
11110	$\text{JUMP}_{\leq}$ i	
11111	JUMP i	$\langle PC \rangle := \langle PC \rangle + [i]$

2. Datenpfade

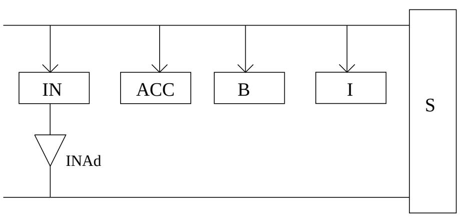
Compute



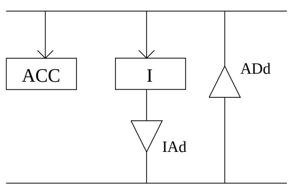
Load



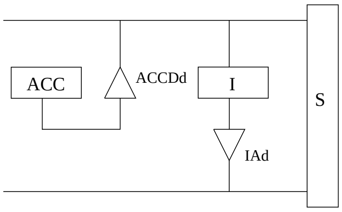
LoadIN



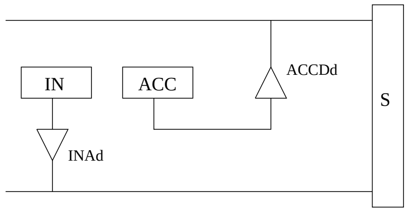
LoadI



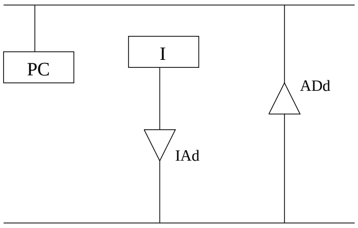
Store



StoreIN



Jump



3. Sign extension

Sei $a = a_n a_{n-1} \ldots a_0 \in \{0,1\}^{n+1}$.
Dann gilt $[a_n a_n a_{n-1} \ldots a_0] = [a]$
Beweis :

$$\begin{aligned}
[a_n a_n a_{n-1} \ldots a_0] &= -a_n 2^{n+1} + \langle a_n a_{n-1} \ldots a_0 \rangle \\
&= -a_n 2^{n+1} + \textstyle\sum_{i=0}^{n} a_i 2^i \\
&= -a_n 2^n + \langle a_{n-1} \ldots a_0 \rangle \\
&= [a]
\end{aligned}$$

Die Schaltung zur Berechnung des Fill-Bits entspricht [Keller/Paul S.308] benutzt aber I15 (statt I23).