BASISPRAKTIKUM

Ausarbeitung

Jörg Gamerdinger und Kim Thuong Ngo

January 14, 2018

Contents

1 Rechenaufgaben

1.1 4+1

Adr.	Befehl	Steuerung		Bus	Register			ALU			Flags
		adr. ctrl.	next adr.	func.	adr. A	adr. B	write	in A	in B	funct. f=	load
0	ADDS R_0 4	-	1	-	0	4	A	-	C	ADDS	-

Steuerung		Bus		Register				ALU			Flags
MAC	NA	WR	EN	AA	AB	WS	WE	Malu IA	Malu IB	Malus	MCH Flags
00	00001	0	0	000	0100	0	0	0	1	0101	0

Erklärung
Die Konstante $B = 4$ wird als 0100 über MRG AB eingegeben. Mit dem Befehl ADDS, wird die Konstante mit dem Inhalt von Register R_0 und 1 addiert, d.h. folgende Rechnung $0 + 4 + 1$ wird ausgeführt. An den LEDs wird 0000 0101 ausgegeben, d.h. das Ergebnis ist 5 und die erwünschte Lösung.

1.2 FF+1

Adr.	Befehl	Steuerung		Bus	Register			ALU			Flags
		adr. ctrl.	next adr.	func.	adr. A	adr. B	write	in A	in B	funct. f=	load
0	ADDS R_0 FF	-	0	-	0	FF	-	-	C	ADDS	x

Steuerung		Bus		Register				ALU			Flags
MAC	NA	WR	EN	AA	AB	WS	WE	Malu IA	Malu IB	Malus	MCH Flags
00	00001	0	0	000	1111	0	0	0	1	0101	1

Erklärung
Das Register FF wird als B eingelesen. Wie im vorigen Versuch a wird der Befehl ADDS verwendet und so $0 + FF + 1$ addiert. Die LEDs geben als Ergebnis 0000 0000 an. Die Frage die aufkommt ist, warum es kein Carry Out gibt. Die Tabelle 5.1 zeigt an, dass der Befehl ADDS ein Carry besitzt, dieser aber invertiert ist. D.h. es gibt kein Carry Out bei dieser Rechnung.

2 MIKROPROGRAMME

Adr.	Befehl	Steuerung		Bus	Register			ALU			Flags
		adr. ctrl.	next adr.	func.	adr. A	adr. B	write	in A	in B	funct. f=	load
0	LD R_4 FF	-	1	-	4	FF	A	-	C	B	-
1	LD R_5 FE	-	2	-	5	FE	A	-	C	B	-
2	LD R_1 (R_4)	-	3	rd	4	1	B	M	-	A	-
3	LD R_2 (R_5)	-	4	rd	5	2	B	M	-	A	-

Steuerung		Bus		Register				ALU			Flags
MAC	NA	WR	EN	AA	AB	WS	WE	Malu IA	Malu IB	Malus	MCH Flags
00	00001	0	0	100	1111	0	1	0	1	1100	0
00	00010	0	0	101	1110	0	1	0	1	1100	0
00	00011	0	1	100	0001	1	1	1	1	0001	0
00	00100	0	1	101	0010	1	1	1	0	0001	0

Erklärung Die Eingabe FF und FE werden als Adresse für Input-Register in den Registern R_4 und R_5 geladen. Mit dem Befehl LD wird auf die ALU die Werte aus F geliefert. Dieser ist am Dateneingang angekoppelt. Die Werte, aus den Memory Adressen FF und FE, werden in den Registern R_1 und R_2 geladen. Deshalb muss der Read Modus 0 sein und Bus Enable 1.

Die Adressen 0-3 dienen zur Initialisierung der Variablen A und B für die folgenden Teilaufgaben (a)-(c).

2.1 (A+B)*2

Adr.	Befehl	Steuerung		Bus	Register			ALU			Flags
		adr. ctrl.	next adr.	func.	adr. A	adr. B	write	in A	in B	funct. f=	load
4	ADD R_2 R_1	-	5	-	2	1	B	R	R	ADD	-
5	ADD R_1 R_1	-	6	-	1	1	A	R	R	ADD	-
6	LD R_3 FF	-	7	-	3	FF	A	-	C	B	-
7	ST (R_3) R_1	-	0	wr	3	1	-	-	R	B	-

Steuerung		Bus		Register				ALU			Flags
MAC	NA	WR	EN	AA	AB	WS	WE	Malu IA	Malu IB	Malus	MCH Flags
00	00101	0	0	010	0001	1	1	0	0	0100	0
00	00110	0	0	001	0001	0	1	0	0	0100	0
00	00111	0	0	011	1111	0	1	0	1	1100	0
00	00000	1	1	011	0001	0	0	0	0	1100	0

Erklärung
Der Wert aus Register 1 wird mit dem Wert aus Register 2 addiert und das Ergebnis in Register 1 gespeichert ($A_{i+1} = A_i + B_i$). Somit ist die Summe aus A und B in Register 1 gespeichert. Die Summe soll mit 2 multipliziert werden, d.h. man addiert das Register mit sich selbst und speichert das Ergebnis wieder in Register 1 ($A_{i+2} = A_{i+1} + A_{i+1}$). Die Adresse vom Output FF wird in das Register 3 geladen. Mit dem letzten Befehl, wird das Ergebnis in Register 1 in das Output Register FF gespeichert.

D.h. das Ergebnis steht am Output FF. Nach Adresse 7 wird Adresse 0 angesteuert, das ist der Jump zum Anfang. Es schafft eine Endlosschleife, um das Programm immer durchlaufen lassen zu können.

2.2 (A-B)/4

Adr.	Befehl	Steuerung		Bus	Register			ALU			Flags
		adr. ctrl.	next adr.	func.	adr. A	adr. B	write	in A	in B	funct. f=	load
4	COM R_2	-	5	-	2	0	A	-	C	NOR	-
5	ADDS R_1 R_2	-	6	-	1	2	A	R	R	ADDS	-
6	LSR R_1	-	7	-	1	0	A	R	R	LSR	-
7	LSR R_1	-	8	-	1	0	A	R	R	LSR	-
8	LD R_3 FF	-	9	-	3	FF	A	-	C	B	-
9	ST (R_3) R_1	-	0	wr	3	1	-	-	R	B	-

Steuerung		Bus		Register				ALU			Flags
MAC	NA	WR	EN	AA	AB	WS	WE	Malu IA	Malu IB	Malus	MCH Flags
00	00101	0	0	010	0000	0	1	0	0	0010	0
00	00110	0	0	001	0010	0	1	0	0	0101	0
00	00111	0	0	001	0000	0	1	0	0	1000	0
00	01000	0	0	001	0000	0	1	0	0	1000	0
00	01001	0	0	011	1111	0	1	0	1	1100	0
00	00000	1	1	011	0001	0	0	0	0	1100	0

Erklärung
Für die Subtraktion muss der Wert B ins Zweierkomplement konvertiert werden. In Adresse 4 wird der Wert aus Register 2 (B) ins Einerkomplement konvertiert. In der nächsten Zeile wird dann folgende Rechnung ausgeführt: $A-B$, weil $A+notB+1$ gerechnet wird. Um durch 4 zu teilen, benutzen wir zweimal ein Bitshift nach rechts, d.h. es werden zwei Nullen links eingeschoben.
Die Adresse des Output Register wird in Register 3 geladen und das Ergebnis im Output Register gespeichert.

2.3 (A AND B)

Adr.	Befehl	Steuerung		Bus	Register			ALU			Flags
		adr. ctrl.	next adr.	func.	adr. A	adr. B	write	in A	in B	funct. f=	load
4	COM R_2	-	5	-	2	2	A	R	R	NOR	-
5	COM R_1	-	6	-	1	1	A	R	R	NOR	-
6	COM R_1 R_2	-	7	-	1	2	A	R	R	NOR	-
7	LD R_3 FF	-	8	-	3	FF	A	-	C	B	-
8	ST (R_3) R_1	-	0	wr	3	1	-	-	R	B	-

Steuerung		Bus		Register				ALU			Flags
MAC	NA	WR	EN	AA	AB	WS	WE	Malu IA	Malu IB	Malus	MCH Flags
00	00101	0	0	010	0010	0	1	0	0	0010	0
00	00110	0	0	001	0001	0	1	0	0	0010	0
00	00111	0	0	001	0010	0	1	0	0	0010	0
00	01000	0	0	011	1111	0	1	0	1	1100	0
00	00000	1	1	011	0001	0	0	0	0	1100	0

Erklärung

Die Werte aus den Registern R_1 und R_2 müssen beide invertiert werden. Danach wird die Funktion NOR für notA und notB verwendet und das Ergebnis in Register 1 gespeichert.
Die Adresse des Output Register wird in Register 3 geladen und das Ergebnis in Output Register gespeichert.

3 MIKROPROGRAMM ZUR KONTOFÜHRUNG

Aufgabe

- Input Register FC: Einzahlung
- Input Register FD: Auszahlung
- Kontostand am Output Register FF
- initialer Kontostand ist 0
- bei continue soll Mikroprogramm laufen (1 Mal)
- letzter Kontostand als Ausgangslage für nächste Berechnung
- Überläufe abfangen (Konto von 0-255)

Adr.	Befehl	Steuerung		Bus	Register			ALU			Flags
		adr. ctrl.	next adr.	func.	adr. A	adr. B	write	in A	in B	funct. f=	load
0	LOOP	x	0/1	-	-	-	-	-	-	-	-
1	LD R_4 FC	-	2	-	4	FC	A	-	C	B	-
2	LD R_5 FD	-	3	-	5	FD	A	-	C	B	-
3	LD R_1 (R_4)	-	4	rd	4	1	B	M	-	A	-
4	LD R_2 (R_5)	-	5	rd	5	2	B	M	-	A	-
5	COM R_2	-	6	-	2	0	A	-	C	NOR	-
6	ADD R_1 R_3	x	8/9	-	1	3	B	R	R	ADD	-
7	-	-	-	-	-	-	-	-	-	-	-
8	ADDS R_3 R_2	x	10/11	-	3	2	A	R	R	ADDS	-
9	LD R_3 FF	-	8	-	3	FF	A	-	C	B	-
10	LD R_7 FF	-	12	-	7	FF	A	-	C	B	-
11	LD R_3 R_0	-	10	-	3	0	A	-	C	B	-
12	ST (R_7) R_3	-	0	wr	7	3	-	-	R	B	-

Steuerung		Bus		Register				ALU			Flags
MAC	NA	WR	EN	AA	AB	WS	WE	Malu IA	Malu IB	Malus	MCH Flags
11	00001	0	0	000	0000	0	0	0	0	0000	0
00	00010	0	0	100	1100	0	1	0	1	1100	0
00	00011	0	0	101	1101	0	1	0	1	1100	0
00	00100	0	1	100	0001	1	1	1	0	0001	0
00	00101	0	1	101	0010	1	1	1	0	0001	0
00	00110	0	0	010	0000	0	1	0	1	0010	0
10	01000	0	0	001	0011	1	1	0	0	0100	0
-	-	-	-	-	-	-	-	-	-	-	-
10	01010	0	0	011	0010	0	1	0	0	0101	0
00	01000	0	0	011	1111	0	1	0	1	1100	0
00	01100	0	0	111	1111	0	1	0	1	1100	0
00	01010	0	0	011	0000	0	1	0	1	1100	0
00	00000	1	1	111	0011	0	0	0	0	1100	0

Erklärung

Der erste Befehl verhindert, dass das Programm läuft. Beim Druck des continue-Taster, kann erst das Mikroprogramm starten, da durch das Drücken die Folgeadresse angesteuert wird. In den Adressen 1-4 werden die Input Register FC und FD als Adresse in Register R_4 und R_5 geladen und dessen Werte in den Registern R_1 und R_2 gespeichert.
Der Wert in R_2 wird konvertiert, um später die Auszahlung auszuführen.
In der nächsten Zeile ist die Einzahlung realisiert, mit dem ADD Befehl. Das Register 3 dient als Speicher für das Konto. Wenn durch die Addition des Kontos und der Einzahlung ein Überlauf auftritt, dann springt man zur Adresse 9, wo Register 3 mit "vollem" Konto, d.h.mit dem Wert 255 geladen wird. Falls es keinen Überlauf gibt, dann wird ganz normal addiert.
Bei der Auszahlung wird der Befehl ADDS verwendet. Analog zur Einzahlung wird hier der Grenzwert 0 abgefangen. Wenn die Summe ins Negative gehen würde, dann springt man zur Adresse 11 und lädt in das Register 3, den Wert 0. Falls durch die Auszahlung das Konto im positiven Bereich bleibt, dann wird die Rechnung durchgeführt.
Zum Schluss wird die Adresse von Output Register FF an die Adresse der Register 7 geladen und der Wert von R_3 in R_7 gespeichert.

4 Versuchsprotokoll

5 KORREKTUR

5.1 AUFGABE 2

Das Flag Signal C_0 zeigt das Carry Out an, d.h. wenn ein arithmetischer Überlauf stattfindet. Normalerweise dürfte C_0 nicht aufleuchten, da die Summe $4+1$ keinen Überlauf hat. Der Grund für das Aufleuchten liegt am Befehl ADDS. Nach Tabelle 5.1 besitzt der Befehl ADDS ein Carry, der aber invertiert. D.h. es leuchtet C_0 bei diesem Befehl auf, da es keinen Überlauf gibt.

5.2 AUFGABE 3

Zeile 1 und 2:
FF und FE werden als Adresse für Input Register in den Registern R_4 und R_5 geladen. Mit dem Befehl LD (F=B) wird der ALU die Werte an FF bzw. FE geliefert.

Zeile 3 und 4:
Um an den Wert von FF und FE ran zu kommen, muss man darauf achten das der Bus aktiv ist (BUS EN = 1) und nicht beschrieben wird (BUS WR = 0). Dann kann man aus der Adresse des Busses den Wert lesen. Dadurch das Malu IA = 1, kann am Eingang A der ALU der Wert weitergereicht werden, der über die MEMDI (Memory Data In) am Eingang des Multiplexers anliegt.
(Verfahren: Input-Register -> Multiplexer -> über ALU gespeichert)
Die Werte werden dann in den Registern R_1 und R_2 gespeichert.

5.3 AUFGABE 3A

Zeile 7:
Die Adresse vom Output FE wird in das Register R_3 geladen.

Zeile 8:
Im letzten Befehl ist der Bus aktiv und wird beschrieben (BUS EN = 1, BUS WR = 1). Der an MEMDO (Memory Data Out) anliegende Wert wird an die eingestellte Zieladresse geschrieben. D.h. das Ergebnis, welcher in Register R_1 gespeichert ist, wird von der MEMA (Memory Adress) ausgewählt, dann an Register R_3 geliefert und so am Output FF ausgegeben.