Dokumentation der Assemblerroutinen

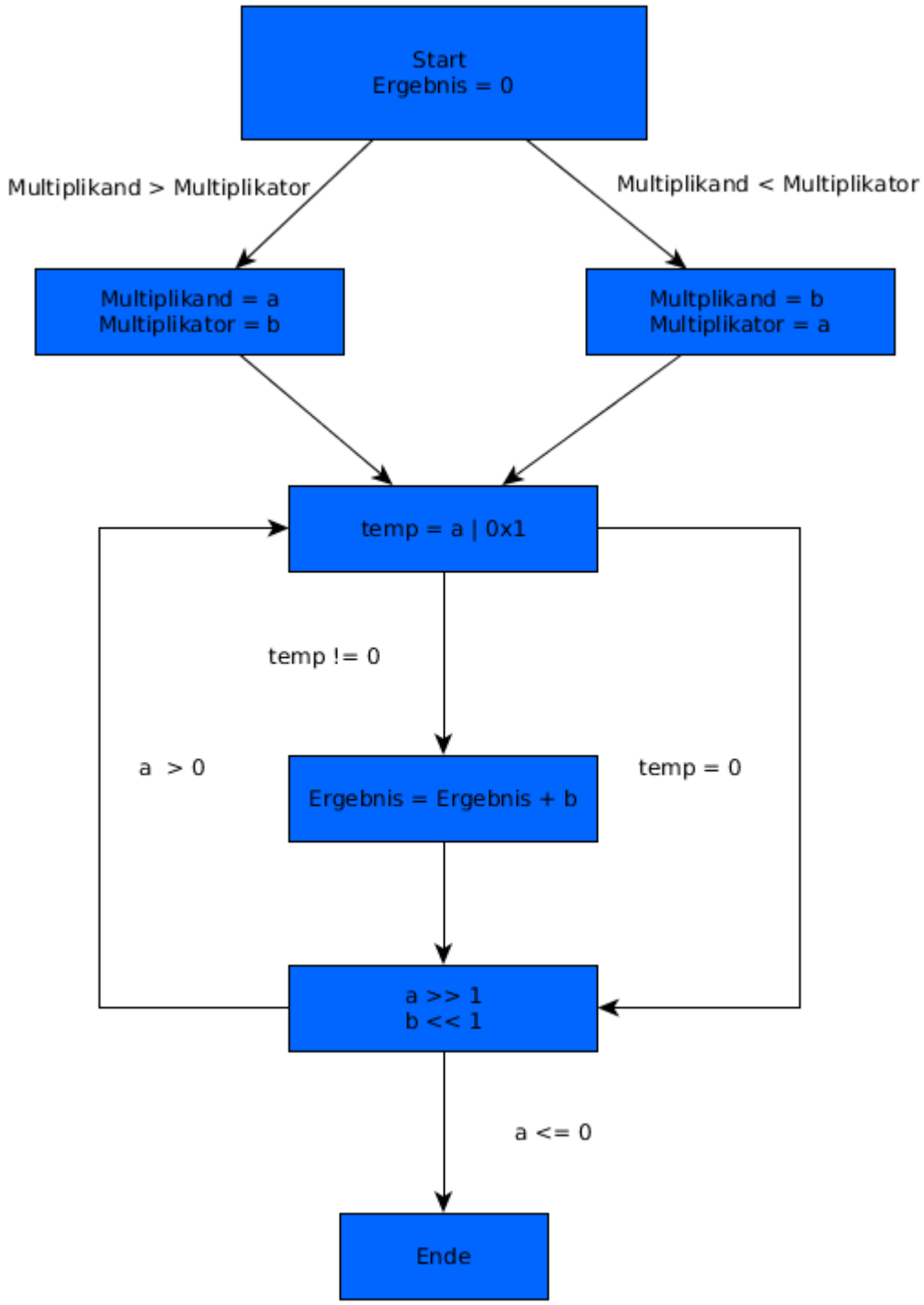
für die Befehle MULT, MULTU, DIV & DIVU

MulU

Zum Multiplizieren, wurde die Methode der „russischen Bauernmultiplikation“ benutzt, die prinzipiell nur ein schriftliches Multiplizieren im Binären darstellt. Auf ein dezimales Beispiel sein an dieser Stelle über den folgenden Linkverwiesen: http://de.wikipedia.org/wiki/Russische_Bauernmultiplikation
Das Grundprinzip geht folgendermaßen:
Beim multiplizieren der Zahlen a und b, schiebt man a immer nach rechts und b immer nach links. Immer dann, wenn a eine ungerade Zahl ist, addiert man b auf das Endergebnis. Dies ist binär immer dann der Fall, wenn das LSB von a eine 1 ist. Dieser Vorgang wird solange widerholt, bis a gleich 0 ist.

Um den Algorhytmus etwas zu beschleunigen, wird zu Anfang geschaut, welche der beiden Zahlen die kleinere ist, und diese wird fortan a genannt. Dies kann zur folge haben, dass der Algorhtymus deutlich weniger Schritte benötigt, da die führende 1 näher am LSB liegt

In einem Flowchart dagestellt sieht der Algorhytmus wie folgt aus:



Dies nun an einem Beispiel

Hierbei bezeichne a = 13 und b =5.

Nach Schritt 0 wird umsortier, daher ist danach a = 5 und b=13

Schritt	a	b	Ergebnis High	Ergebnis Low
0	1101	0101	-	0000
1	0101	0000 1101	-	0000 1101
2	0010	0001 1010	-	0000 1101
3	0001	0011 0100	-	0100 0001

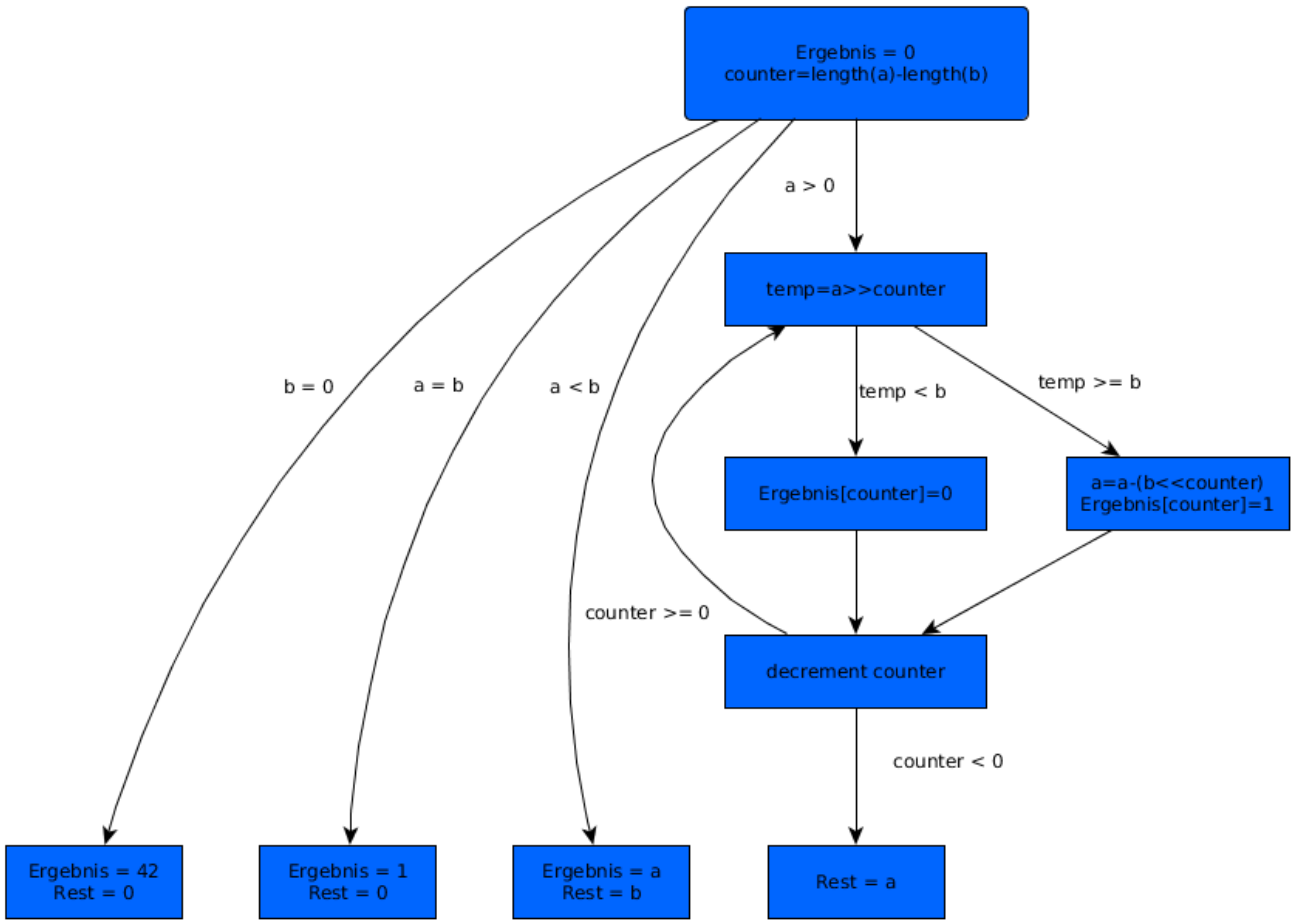
Die Zahl 0b1000001 ist dezimal 65. Das Ergebnis ist offensichtlich richtig, und sehr einfach zu realisieren. Schwierigkeiten, macht hier lediglich, dass wir später bei der Addition, zwei 64-Bit Zahlen addieren, bzw eine 64-Bit zahl shiften müssen. Hierzu muss bei der Addition von Hand der Überlauf bei der Addition der unteren Zahlenhälfte geachtet werden und dieser von Hand auf die obere Hälfte addiert werden. Bei Shift muss das MSB der unteren Hälfte gespeichert und später als neues LSB oder oberen Häflte angefügt werden.

Mul

Die Vorzeichebehaftete Multiplikation läuft genauso wie die ohne Vorzeichen ab. Dazu werden negative Werte über eine Subtraktion von null ins positive überführt und die beiden Vorzeichen werden Verodert. Soll das Ergebnis nun negativ sein, wird das positive Ergebnis noch mit Hilfe des Zweierkomplements umgewandelt.

DIVU

Bei der Division wurde auf die klassische, schriftliche Division zurückgegriffen, die binär umgesetzt wurde, da diese ohne große Lookup-Table auskommt (siehe zum Vergleich die SRT-Division) und keine weitere Hardware benötigt (siehe zum Vergleich die Goldschmidt-Division, die eine sukzessive Aproximation der Lösung auf eine Multiplikation zurückführt.



Dies sei an Hand eines kurzen Beispiel erklärt:

Es werde 13 durch 2 geteilt, binär also 1101 / 0010. Der Counter beginnt mit der Differenz der Stellen der beiden Zahlen, die diese um ihre erste 1 versetzt sind. In diesem Fall also 2. betrachten wir zunächst den Fall, dass eine eigentliche Division stattfindet, das Ergebnis also nicht direkt per Konvention o.Ä. klar ist.

Hierbei bezeichne a = 13 und b =2.

Schritt	a	b	counter	temp	Rest	Ergebnis
0	1101	0010	-	-	-	0000
1	1101	0010	10	-	-	0000
2	0101	0010	10	0011	-	0100
3	0101	0010	01	0011	-	0100
4	0101	0010	01	0010	-	0100
5	0001	0010	01	0010	-	0110

6	0001	0010	00	0010	-	0110
7	0001	0010	00	0010	0001	0110

Das binäre Ergebnis 0110, also in dezimal 6 und der Rest 1 entsprechen dem Erwarteten.

Im Falle eine Divisors, der größer als die Divident ist, ist per MIPS Konvention festgelegt, dass, da keine Zahlen kleiner 1 abgebildet werden können, die Übergabeparameter wieder zurück übergeben werden.

Im Falle der Division durch 0, welcher nicht über eine MIPS Konvention geregelt ist, wird die Antwort auf die Frage „nach dem Leben, dem Universum und dem ganzen Rest“[1] als Lösungsvorschlag zurückgegeben.

DIV

Bei der Division mit Vorzeichen muss nun lediglich die vorzeichenbetrachtung ergänzt werden. Bei identischen Vorzeichen ist das Resultierende Ergebnis positiv, andernfalls negativ. Dies wird analog zur Vorzeichenbehandlung der Multiplikation umgesetzt. Das Vorzeichen des Restes ist analog zu dem des Dividenden.

1 Siehe Douglas Adams, Per Anhalter durch die Galaxis