Power of Two	Decimal Value	Hexidecimal Value
2^0	1	x 1
2^1	2	x 2
2^2	4	x 4
2^3	8	x 8
2^4	16	x 10
2^5	32	x 20
2^6	64	x 40
2^7	128	x 80
2^8	256	x 100
2^9	512	x 200
2^{10}	1,024	x 400
2^{11}	2,048	x 800
2^{12}	4,096	x 1000

Zuerst wird Y_0 inkl. Übertrag via **ADDS** berechnet. Anschließend folgt die Berechnung von Y_1 via **ADC** oder **ADCS**. 64-Bit Addition / Subtraktion

Assembler #081

```
                            ; berechne (R0, R1) + (R2, R3)
                            ; R0 und R2: niederwertige Wörter (Bits 0...31)
                            ; R1 und R3: höherwertige Wörter (Bits 31...63)
ADDS    R4, R0, R2          ; addiere die niederwertigen Wörter
ADC     R5, R1, R3          ; addiere die höherwertigen Wörter + C-Flag
```

Der Assembler–Code für die 64–Bit–Subtraktion ergibt sich analog.

Assembler #082

```
                            ; berechne (R0, R1) - (R2, R3)
                            ; R0 und R2: niederwertige Wörter (Bits 0...31)
                            ; R1 und R3: höherwertige Wörter (Bits 31...63)
SUBS    R4, R0, R2          ; subtrahiere die niederwertigen Wörter
SBC     R5, R1, R3          ; subtrahiere die höherwertigen Wörter
```

Diese Vorgehensweise lässt sich problemlos auf Operanden mit $n \cdot 32$ Bits erweitern.

Soll ein Wert vom Datentyp `int64_t` und ein Wert vom Datentyp `int32_t` addiert werden, so muss das Vorzeichen korrekt erweitert werden.

Assembler #083

```
                            ; 64-Bit-Wert x (int64_t) in R0 und R1
                            ; 32-Bit-Operand a (int32_t) in R2
                            ; x ← x + a
  ADDS    R0, R0, R2        ; addiere a zum niederwertigen Wort
  MOV     R2, R2, ASR #31   ; -1 wenn a < 0, sonst 0
  ADC     R1, R1, R2        ; höherwertiges Wort anpassen
```

Im Gegensatz dazu der Code für einen Operanden vom Typ `uint32_t`.

Assembler #084

```
                            ; 64-Bit-Wert x (sint64_t) in R0 und R1
                            ; 32-Bit-Operand a (uint32_t) in R2
                            ; x ← x + a
  ADDS    R0, R0, R2        ; addiere a zum niederwertigen Wort
  ADC     R1, R1, #0        ; addiere Übertrag
```