# Das kann die ALU:

- Zwei Register miteinander Addieren (ADD), es kann ein vorher ermittelter Überlauf (das Carry aus dem SREG) miteinbezogen werden (ADC).  
  
- Analog dazu zwei Register subtrahieren, ggf. auch hier mit Carry (SUB bzw. SBC)  
  
- Von einem Register eine Konstante abziehen, auch hier ggf. mit Carry (SUBI, SBCI)  
  
- Auf ein Doppelregister (16bit) eine Konstante addieren (ADIW) bzw. von dort subtrahieren (SBIW)  
  
- Zwei Register bitweise logisch verknüpfen (AND, OR, EOR)  
  
- Ein Register mit einer Konstante bitweise logisch verANDen (ANDI) bzw. verORen (ORI)  
  
- Von einem Register das Einer- (COM) bzw. Zweierkomplement (NEG) bilden  
  
- Ein Register inkrementieren (INC) bzw. dekrementieren (DEC)  
  
- Zwei Register multiplizieren, und zwar beide unsigned (MUL), beide signed (MULS) oder eins singned und eines unsigned (MULSU), analog dazu fraktionelle Multiplikation (FMUL, FMULS, FMULSU)

**Hierher gehören eigentlich auch noch folgende Bit-Funktionen, die ALU kann:**  
  
- Die Bits eines Registers logisch nach links (LSL) oder rechts (LSR) schieben, dabei rutscht eine null auf die leere Stelle, das rausgeschobene Bit fällt ins Carry  
  
- Statt der null kann auch das Carry auf die leere stelle gerollt werden (ROL bzw. ROR)  
  
- Die Bits eines Registers arithmetisch nach rechts schieben (ASR, Integer Division durch zwei)

**Und der Vollständigkeit halber noch die drei Vergleiche - das sind in Wirklichkeit nur Subtraktionen ohne Zurückschreiben des Ergebnisses. Trotzdem wird das SREG manipuliert:**  
  
- CP vergleicht (subtrahiert) zwei Register  
  
- CPC subtrahiert zusätzlich das Carry  
  
- CPI subtrahiert eine Konstante vom Register  
  
Streng genommen haben wir damit alle Instruktionen, in denen die ALU durch eine Operation Bits im SREG manipuliert.  
(hinter SBR verbirgt sich in Wirklichkeit ORI, hinter CBR ANDI, CLR ist eigentlich ein EOR, SER ein LDI (also gar keine Rechenoperation, deswegen kommt auch das Zero nicht), TST ist ein AND)