# TOY OpCodes

1. Geben Sie 5 Anweisungen an (mit unterschiedlichen OpCodes), die No-Ops sind. Schließen Sie dabei Fälle aus, bei denen das Zielregister 0 ist.
2. Geben Sie ein No-Op im 2. Format an.
3. Geben Sie 6 Möglichkeiten an, den Inhalt von Register B Register A zuzuweisen.

4. Es gibt kein *bitweises Not* in TOY. (In Java: ~.) Beschreiben Sie, wie man RB <- ~RA mit Hilfe einer Sequenz von 3 TOY-Anweisungen berechnen kann. (Bei RB sind alle Bits von RA einmal umgedreht).
5. Es gibt kein *bitweises Or* in TOY. (In Java: |.) Beschreiben Sie, wie man RC <- RA | RB mit Hilfe einer Sequenz von 3 TOY-Anweisungen berechnen kann. *(Das i-te Bit von RC soll genau dann 1 sein, wenn das i-te Bit von RA oder das i-te Bit von RB 1 ist.)*
6. Es gibt kein *bitweises Nor* in TOY (oder Java). Beschreiben Sie, wie man RC <- RA nor RB mit Hilfe einer Sequenz von TOY-Anweisungen berechnen kann. *(Das i-te Bit von RC soll genau dann 0 sein, wenn das i-te Bit von RA oder das i-te Bit von RB 1 ist.)*
7. Es gib kein *bitweises Nand* in TOY (oder Java). Beschreiben Sie, wie man RC <- RA nor RB mit Hilfe einer Sequenz von TOY-Anweisungen berechnen kann. *(Das i-te Bit von RC soll genau dann 0 sein, wenn das i-te Bit von RA und das i-te Bit von RB 1 sind.)*
8. Zeigen Sie, dass der *subtract*-Operator redundant ist. D.h., Sie sollen beschreiben, wie man
   RC <- RA – RB mit Hilfe einer Sequenz von TOY-Anweisungen berechnen kann, ohne den Opcode 2 zu benutzen.