x86 Assembler Übung: Erweitertes `popcnt`

Klaus Kusche

Erweitere das **`popcnt`**-Beispiel.

Betrachte dazu wieder das gegebene Hauptprogramm:

Es übergibt der Funktion als *zweiten Parameter* einen *Pointer auf eine Struktur*
mit zwei **`int`**'s, in die die Funktion die *Position des ersten und des letzten 1-Bits*
speichern soll.

Wenn der Wert **`0`** ist (d.h. keine 1-Bits enthält),
dann soll in beiden Struktur-Membern **`-1`** gespeichert werden.

Hinweise:

- Schau Dir auf den Folien an, wie man *Speicherzugriffe* macht!
- Die beiden Befehle zum Suchen des ersten und letzten 1-Bits heißen **`bsf`** und **`bsr`**.

  Beide können ihr Ergebnis *nur in einem Register* speichern.
  Du wirst das Ergebnis daher danach mit einem zweiten Befehl
  aus dem Register in die Struktur im Speicher *kopieren* müssen.

  Welche Register darfst du zur Zwischenspeicherung verwenden?
  Wenn du eines der „neuen" Register verwendest:
  Vergiss nicht auf die *richtige Länge* (wir reden von 32-bit-Werten!).
- Du brauchst *kein* separates **`test`** oder **`cmp`** für den *Sonderfall*:
  `popcnt` setzt das Zero-Flag, wenn Input und Ergebnis **`0`** sind.
- Im Sonderfall kannst du die Konstante **`-1`** mit einem **`mov`**
  direkt in der Struktur speichern (ohne Umweg über ein Register).

  Aber da der Assembler weder aus dem Speicherzugriff
  noch aus dem **`-1`** die *Operanden-Länge* erkennen kann,
  musst du diese beim Speicherzugriff *explizit angeben* (siehe Folien!).