HEINRICH-HEINE-UNIVERSITÄT DÜSSELDORF
INSTITUT FÜR INFORMATIK
LEHRSTUHL FÜR RECHNERNETZE
PROFESSOR DR. M. MAUVE

HEINRICH HEINE
UNIVERSITÄT DÜSSELDORF

# Grundlagen der Technischen Informatik

Übungsblatt 12

---

**Ankündigungen:** Dieses Übungsblatt ist das letzte Übungsblatt in Informatik 2.
Insgesamt gab es also **12 Übungsblätter** mit je 16 Punkten.
Die **Zulassungsgrenze** wurde von 60% der Punkte auf **50%** gesenkt.

**Abgabefrist:** 10.07.2013

**Ansprechpartner:** Der Tutor ihrer Übungsgruppe

**Geben Sie zu jeder Aufgabe Ihren Lösungsweg in eigenen Worten an!**

---

**Aufgabe 12.1** *Mic: ALU* (2 Punkte)

Die Ansteuerung der einzelnen Funktionen der ALU (Arithmetic Logic Unit) der MIC ist auf dem Foliensatz der Vorlesung, Kapitel 8 Folie 9, tabellarisch dargestellt.

1. Es gibt unterschiedliche Möglichkeiten, die Funktion $f(A,B) = \overline{B}$ mit der ALU zu berechnen. Geben Sie vier unterschiedliche Belegungen für die Steuersignale $F_0, F_1, ENA, ENB, INVA$ an, die die Funktion $f$ berechnen. Gehen Sie davon aus, dass $INC = 0$ gesetzt ist.

   **Lösungsvorschlag:** (1 Punkt)

   | $F_0$ | $F_1$ | $ENA$ | $ENB$ | $INVA$ |
   |---|---|---|---|---|
   | 1 | 0 | 0 | 1 | 0 |
   | 1 | 0 | 0 | 1 | 1 |
   | 1 | 0 | 1 | 1 | 0 |
   | 1 | 0 | 1 | 1 | 1 |

2. Geben Sie die Funktion $g(A,B)$ an, die die ALU berechnet, wenn

$$F_0 = 0, F_1 = 0, ENA = 1, ENB = 1, INVA = 1, INC = 0.$$

**Lösungsvorschlag:** (1 Punkt)

$$g(A,B) = \overline{A} \text{ AND } B$$

**Aufgabe 12.2** *Mic: Programm* (3 Punkte)
Geben Sie eine Folge von Mikroinstruktionen für die Mic-1 an (in MAL, angemessen kommentiert), die Folgendes erreicht: Von der in CPP gespeicherten Adresse soll 1 Wort (d. h. 4 Byte) gelesen und von dem in TOS gespeicherten Wert subtrahiert werden. Das Ergebnis soll in TOS abgelegt und außerdem an die in SP gespeicherte Speicheradresse geschrieben werden.

**Lösungsvorschlag:** (3 Punkte)

```
MAR = CPP; rd;   // Leseoperation von Adresse in CPP; Daten sind 2 Takte später da
MAR = SP;        // Adresse aus SP in MAR legen
H = MDR;         // Daten aus dem Speicher sind jetzt angekommen, nach H kopieren
TOS = MDR = TOS - H; wr; // rechnen, Ergebnis sowohl nach TOS als auch in den RAM
```

**Aufgabe 12.3** *Mic: Shifter* (5 Punkte)

1. Wie kann das Schieben um ein Bit nach links prinzipiell realisiert werden? Welche Einschränkungen gibt es dabei?

   **Lösungsvorschlag:** (1 Punkt)

   Durch ein Schieben von 8 Bits nach links und danach siebenmal nach rechts. Der Ausgangswert sollte aber nicht mehr Bits benötigen als $32-7=25$. Sonst wird das Ergebnis in den oberen sieben Bits verfälscht.

2. Erklären Sie, wie Bitmasken in einem Mikroprogramm erzeugt werden können. Welche Rolle spielen dabei Schiebeoperationen? Identifizieren Sie Bitmasken für das *most* und *least significant bit* und realisieren Sie diese durch MAL-Anweisungen.

   **Lösungsvorschlag:** (0,5 + 0,5 + 1 Punkte)

   Prinzipiell kann man Bitmasken durch addieren von Einsen zu einem Ausgangswert erreichen, die man geeignet shiftet, bis die Bitmaske die gewünschte Form erhält.

   - LSB:

     ```
     H = 1;              // Eine Eins erzeugen, fertig.
     ```

   - MSB:

     ```
     // Eine Eins erzeugen und um insgesamt 31 verschieben
     H = 1 << 8;         // Eine Eins erzeugen
     H = H << 8;         // ...verschieben...
     H = H << 8;         // ...verschieben...
     H = H >> 1;         // jetzt einmal nach rechts verschieben
     H = H << 8;         // und nochmal nach links verschieben
     ```

3. In Mikroprogrammen sind Schleifen oft sehr hilfreich. Wie kann eine Schleife in MAL prinzipiell durch einen numerischen Zähler realisiert werden, wie durch eine (1-)Bitmaske?

   **Lösungsvorschlag:** (2 Punkte)

   Beiden Varianten ist gemein, dass Sie durch Sprunginstruktionen gesteuert werden. Am Beginn einer Schleife wird eine Bedingung geprüft. Ist diese wahr, wird aus der Schleife 'herausgesprungen', also hinter die letzte Anweisung der Schleife. Diese letzte Anweisung ist typischerweise ein unbedingter Sprung zurück zur Prüfung der Schleifenbedingung. Die Bedingung kann für beide Varianten eine Prüfung einer Variablen auf Gleichheit mit Null sein. Innerhalb der Schleife muss dann die Variable so verändert werden, dass eine Beendigung der Schleife nach der gewünschten Zahl Iterationen ermöglicht wird.

   - Numerischer Zähler: Eine Variable wird z.B. nach H kopiert und dann in der ALU durchgeschleift. Ist das Z-Signal der ALU gesetzt, war der Wert in H Null. Dann wird aus der Schleife gesprungen. Ansonsten wird die Schleife abgearbeitet, bis ein Sprung zurück zur Prüfung führt. Bis dahin muss in H (wieder) der Zähler stehen, welcher zudem dekrementiert, also um eins reduziert, wurde.
   - Bitmaske: Ganz ähnlich zum numerischen Zähler wird hier eine Variable, wieder z.B. in H, benutzt. Diese besteht in der Bit-Repräsentation aus Nullen und maximal einer Eins. Der zweite Unterschied zum numerischen Zähler ist, dass statt der Subtraktion eine Schiebeoperation um eins nach rechts den Wert der Variablen verringert.

**Aufgabe 12.4** *Mic: Control Store* (6 Punkte)

Betrachten Sie für die Unteraufgaben das nachfolgende MAL-Programm:

```
    MAR = SP = SP + 1;
    N = TOS; if (N) goto L2; else goto L1;
L1  Z = TOS; if (Z) goto L2; else goto L3;
L2  TOS = MDR = 1; wr; goto Main1;
L3  TOS = MDR = 0; wr; goto Main1;
```

1. Beschreiben Sie, was das Mikroprogramm tut (bzw. was es tun soll).

   **Lösungsvorschlag:** (2 Punkte)

   Das Programm soll das Datenwort, welches oben auf dem Stack (und somit auch in TOS) liegt, mit dem Wert 0 vergleichen. Gilt $TOS \leq 0$, so soll eine 1, andernfalls eine 0 auf den Stack gelegt werden.

2. Warum funktioniert es so nicht? (Hinweis: Beachten Sie, wie die einzelnen Anweisungen im Control Store platziert werden müssen.)

**Lösungsvorschlag:** (2 Punkte)

Das Problem liegt in den Sprunganweisungen: Bei einem bedingten Sprung wird das neunte Bit der Sprungadresse abhängig vom Resultat des Vergleiches auf 0 oder 1 gesetzt:
`MPC[8] = (JAMZ AND Z) OR (JAMN AND N) OR NEXT_ADDRESS[8]`

Das bedeutet, dass die Sprungmarken des `then` und `else`-Zweiges um 256 Byte im Control Store auseinander liegen. Dies bedeutet, dass eine Sprungmarke immer nur mit genau einer anderen Sprungmarke kombiniert werden kann. Im obigen Quelltext wird aber die Sprungmarke `L2` einmal mit `L1` und einmal mit `L3` kombiniert.

3. Beschreiben Sie eine mögliche Lösung und ändern Sie das Programm entsprechend.

**Lösungsvorschlag:** (2 Punkte)

Der Fehler kann behoben werden, indem eine weitere Sprungmarke eingeführt wird, von der aus ein unbedingter Sprung nach `L2` vollzogen wird. Der korrigierte Quelltext lautet also:

```
    MAR = SP = SP + 1;
    N = TOS; if (N) goto L2; else goto L1;
L1  Z = TOS; if (Z) goto L4; else goto L3;
L2  TOS = MDR = 1; wr; goto Main1;
L3  TOS = MDR = 0; wr; goto Main1;
L4  goto L2;
```

**Wir wünschen Ihnen viel Erfolg beim C-Projekt und der Klausur!**