



# Rechnerorganisation

## SS 2005

### Übungsblatt Nr. 5



Prof. Dr. Robert Tolksdorf, AG Netzbasierte Informationssysteme  
Freie Universität Berlin

**Ausgabe am 17.06.2005 — Abgabe spätestens 01.07.2005, 12:00 Uhr**

Bitte bei der Abgabe alle Namen/Matr.Nr. der Gruppe, NUMMER DER ÜBUNG/TEILAUFGABE und DATUM auf den Lösungsblättern **nicht vergessen!** Darauf achten, dass die Lösungen beim richtigen Tutor/der richtigen Tutorin abgegeben werden.

**Zu spät abgegebene Lösungen werden nicht mehr berücksichtigt!**

---

## 1. Aufgabe: Klausurvorbereitung (0 Punkte – aber sinnvoll!)

Schauen Sie sich eine alte Klausuren zur Vorbereitung an und versuchen Sie diese selbstständig zu lösen: [http://nbi.inf.fu-berlin.de/lehre/05/V\\_RO/testfragen.pdf](http://nbi.inf.fu-berlin.de/lehre/05/V_RO/testfragen.pdf). **Achtung: Die kommende Klausur ist nicht identisch vom Aufbau her, es werden auch Freitextaufgaben etc. enthalten sein.**

## 2. Aufgabe: Virtueller Speicher (3 Punkte)

Ein Computer habe einen virtuellen Adressraum mit 128 Seiten, aber lediglich 4 Seitenrahmen. Anfangs sei der Speicher leer. Ein Programm referenziere die virtuellen Seiten in folgender Reihenfolge:

0, 7, 2, 9, 7, 4, 5, 2, 9, 2, 4

Welche Referenzen verursachen einen Seitenfehler bei LRU als Ersetzungsstrategie?

- A. 5.
- B. 6.
- C. 8.
- D. 9.

Welche Referenzen verursachen einen Seitenfehler bei FIFO als Ersetzungsstrategie?

- E. 5.
- F. 6.
- G. 8.
- H. 9.

## 3. Aufgabe: Heaps (12 Punkte)

Implementieren Sie IA-32-Assembler-Routinen zur Verwaltung eines Heaps. Sie finden ausführliche Hinweise zu einem Heap unter: <http://www.iti.fh-flensburg.de/lang/algorithmen/sortieren/heap/heap.htm>. Programmieren Sie den Heap *ohne* die Verwendung von Zeigern, lediglich durch ein Feld (Ausnahme: der Zeiger auf das Feld). Ihr Heap sollte mit bis zu 256 Elementen klar kommen. Implementieren Sie folgende Funktionen:

1. `create_heap`: Erzeugen eines neuen Heaps; Eingabe: erstes Element, Zeiger auf den freien Speicher
2. `insert_heap`: Einfügen eines neuen Elements in den Heap; Eingabe: Zeiger auf den Heap, neues Element
3. `sort_heap`: Sortieren eines noch nicht sortierten Feldes in einen Heap; Eingabe: Zeiger auf die zu sortierende Liste
4. `remove_top_heap`: Entfernen der Wurzel, Eingabe: Zeiger auf den Heap; Ausgabe: Wurzelelement

## 5. Aufgabe: Betriebssystem und Speicher (3 Punkte)

Welche Aussagen zum Thema Betriebssysteme und Assembler treffen zu? Begründen Sie jeweils durch Beispiele/Gegenbeispiele/Erläuterungen!

1. Speicherschutzmechanismen eines Betriebssystems müssen durch den Prozessor unterstützt werden.
2. Assembler-Befehle können direkt von einem Prozessor ausgeführt werden.
3. Die Auslagerung und Einlagerung von Seiten geschieht durch das Betriebssystem.
4. Die MMU ist für Einlagerung oder Auslagerung von Segmenten zuständig.
5. Programmierer müssen den Ort eines Programms im Speicher schon bei der Programmierung festlegen.
6. Betriebssystemfunktionen müssen regelmäßig aufgerufen werden, um zwischen Prozessen umzuschalten.
7. Auch Betriebssysteme können bei einem Speicherengpass (zu wenig RAM) vollständig auf die Festplatte ausgelagert werden.
8. DLLs müssen an die von den benutzenden Programmen referenzierten Speicherbereiche geschrieben werden.

## 5. Aufgabe: Booth-Algorithmus (12 Punkte)

Implementieren Sie eine IA-32-Assembler-Routine zur Multiplikation zweier 32-Bit-Zahlen in 2er-Komplement-Darstellung nach dem Booth-Algorithmus. Die zugehörige Funktion sollte wie folgt aufrufbar sein:

`int mult_booth(int a, int b)` und das Produkt von a und b als Rückgabewert liefern. Die folgende Abbildung zeigt den Algorithmus schematisch.

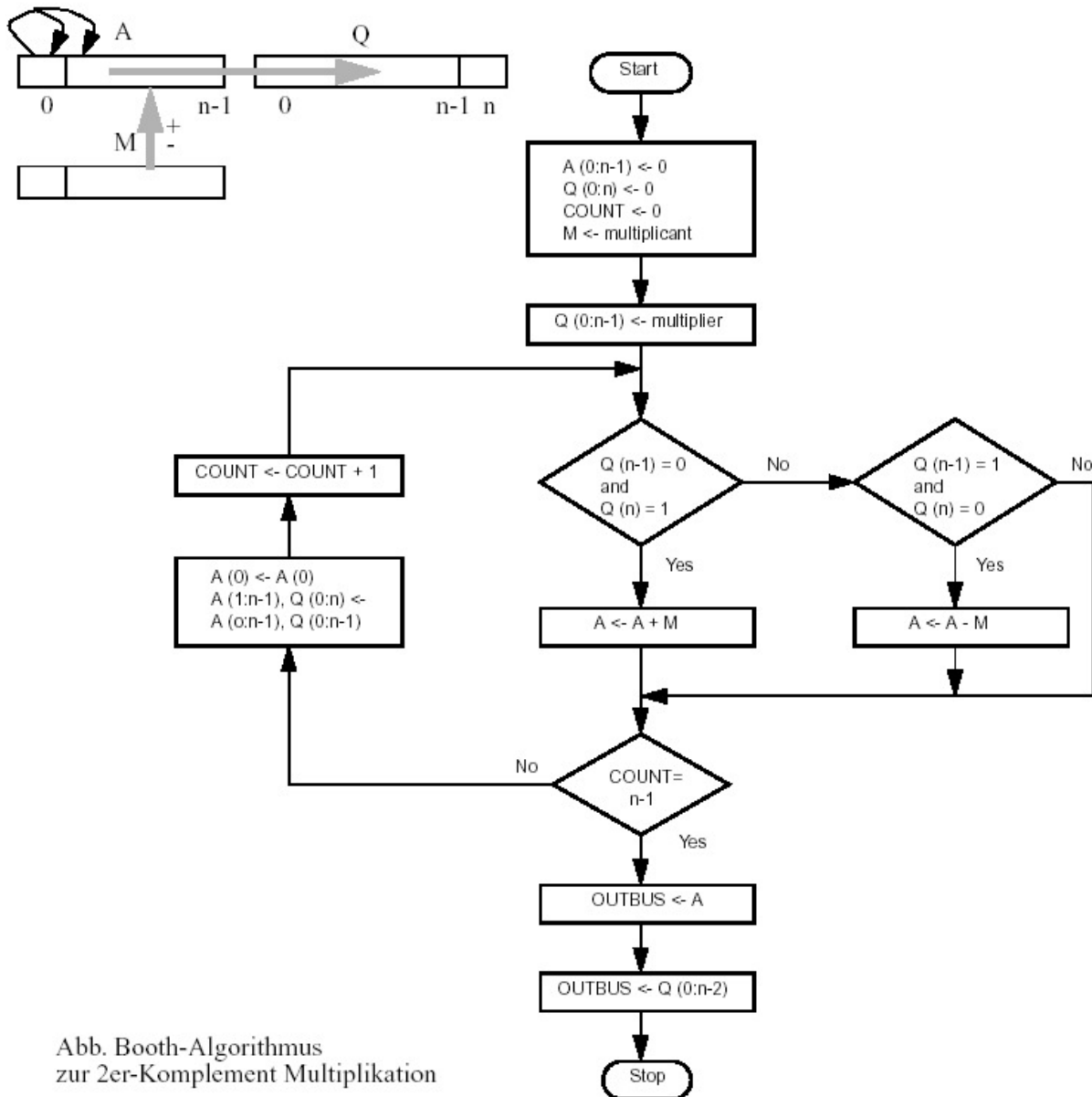


Abb. Booth-Algorithmus zur 2er-Komplement Multiplikation