FAKULTÄT FÜR INFORMATIK
TECHNISCHE UNIVERSITÄT MÜNCHEN

Lehrstuhl für Rechnertechnik und Rechnerorganisation
Prof. Dr. Martin Schulz

Einführung in die Rechnerarchitektur

Wintersemester 2017/2018

Tutorübung 6

27.11–01.12.2017

---

1. (Klausuraufgabe WS05) Mustersuche

   Viele digitale Datenströme nutzen bestimmte, aufeinanderfolgende Bytemuster zum einfachen Auffinden bestimmter Informationen. MPEG-Videoströme verwenden z. B. die Bytefolge „0x00 0x00 0x01“, um den Anfang einer Informationsgruppe zu markieren.

   Schreiben Sie das 80386-Unterprogramm `find_marker`, das so ein Muster findet und den folgenden Anforderungen genügt:

   - Das zu findende Bytemuster ist „0x00 0x00 0x01“.
   - Die Startadresse des zu durchsuchenden Speicherbereichs stehe in `esi`.
   - Die Länge (in Bytes) des Speicherbereichs stehe in `edx`.
   - Wird das Muster gefunden, kehrt das Unterprogramm mit der Startadresse des Musters in `eax` zurück.
   - Wird das Muster nicht gefunden, soll in `eax` der Wert -1 zurückgegeben werden.
   - Die Register dürfen alle verändert werden.
   - Eine besonders hohe Effizienz des Programms ist nicht erforderlich.

2. (Klausuraufgabe WS05) Suche in einem sortierten Baum

   Es soll eine zu einem bestimmten Schlüsselwert gehörende Information in einer baumartigen Speicherstruktur gesucht werden. Die Struktur eines Baumknotens sieht dabei folgendermaßen aus:

Startadresse	Schlüsselwert (32-Bit-Zahl)
Startadresse + 4	Zeiger auf Information
Startadresse + 8	Zeiger auf linken Teilbaum
Startadresse + 12	Zeiger auf rechten Teilbaum

   Über die Zeiger auf die linken bzw. rechten Teilbäume wird der Baum aufgebaut. Ist ein Zeiger 0, ist keine Verzweigung in diese Richtung vorhanden. Der Baum ist sortiert, d. h. der Schlüsselwert eines Knotens ist immer größer als alle Schlüsselwerte seines linken Teilbaums und immer kleiner als alle Schlüsselwerte seines rechten Teilbaums.

   (a) Mit welchem 80386-Befehl kann man die Startadresse des rechten Teilbaumes in das Register `ebx` bekommen, wenn die Knotenadresse im Register `ebx` steht?

(b) Schreiben Sie ein 80386-Unterprogramm `find`, das den zu einem Schlüsselwert gehörenden Informationszeiger zurückgibt. Der zu suchende Wert stehe im Register `ecx`, die Startadresse der Baumwurzel (des obersten Knotens) im Register `ebx`. Der Informationszeiger soll im Register `eax` stehen. Wird der Schlüsselwert nicht gefunden, soll 0 im Register `eax` zurückgegeben werden.

Hinweise:

- Achten Sie darauf, dass unter keinen Umständen auf die Speicheradresse 0 zugegriffen wird.
- Es sind alle Register veränderbar.
- Vermeiden Sie Rekursion, die den Stack nutzt.

3. (Klausuraufgabe WS05) Bitspielerei

   Schreiben Sie ein 80386-Unterprogramm, das ein Bit im Register `eax` löscht. Die Bitnummer wird im Register `ecx` übergeben, dabei entspricht 0 dem niederwertigsten Bit und 31 dem höchstwertigen Bit.

   Folgende Befehle dürfen dabei **nicht** verwendet werden: `and, or, xor, not, add, sub, mul, div, jcc`.

4. (Klausuraufgabe WS96) Skalarprodukt

   Das Skalarprodukt auf Vektoren a und b mit je n Elementen ist wie folgt definiert:

$$ab = \sum_{i=0}^{n-1} a_i \cdot b_i$$

   Schreiben Sie das Unterprogramm `smul` für vorzeichenbehaftete 32-Bit-Zahlen in 80386 Assembler. Die Startadressen der Felder a bzw. b werden in `eax` bzw. `ebx` übergeben, die Elementanzahl n wird im Register `ecx` übergeben. Das 32-Bit-Ergebnis soll in Register `eax` abgelegt werden. Überläufe etc. können Sie ignorieren. Register `ebx` eund `ecx` müssen nach der Ausführung unverändert sein.

5. (Klausuraufgabe WS04) Taschenrechner

   In den folgenden Teilaufgaben soll eine Art Taschenrechner entwickelt werden. Im Unterschied zu normalen Geräte verarbeitet dieser seine Eingaben aber in der sogenannten *Umgekehrt Polnischen Notation (UPN)*. Bei UPN werden die Eingaben auf einem Stack gespeichert. Die Rechenoperationen verarbeiten bzw. verbrauchen die obersten Elemente und geben das Ergebnis wieder auf den Stack.

   Die Beispieleingabe: „1 2 2 * + 3 +“ ergibt nach der Berechnung den Wert 8 ($2 \cdot 2 + 1 + 3$), der auf dem Stack übrig bleibt.

   Um die Stacks für Rechenwerte und Rücksprünge nicht zu vermischen, müssen die grundlegenden `push` bzw. `pop`-Operationen für den Rechenstack auf dem 80386 nachgebildet werden. Der Zeiger dazu sei im Register `esi`; er zeigt auf die nächste freie Stackspeicherzelle. Die Wachstumsrichtung sei wie die des normalen Systemstacks.

   (a) Schreiben Sie in 80386-Assembler das Unterprogram `push_value`, das einen 32-Bit Wert, der in `eax` übergeben wird, auf den Rechenstack legt. Die Register `eax` bis `edx` sollen dabei unverändert bleiben.

   (b) Schreiben Sie in 80386-Assembler das Unterprogram `pop_value`, das einen 32-Bit Wert vom Rechenstack nimmt und im Register `eax` zurückgibt. Die Register `ebx` bis `edx` sollen dabei unverändert bleiben.

(c) Die Addition nimmt zwei Werte vom Rechenstack, addiert sie und gibt das Ergebnis wieder auf den Rechenstack. Schreiben Sie unter Zuhilfenahme der Unterprogramme `push_value` und `pop_value` das Unterprogramm `add_values`, das diese Addition ausführt.

(d) Der Taschenrechner bekommt seine Eingabe als ein Folge von hintereinanderliegenden Bytes, die einzelne Zeichen darstellen. Dabei entspricht die Bedeutung der Zeichenwerte von 48 bis 57 (dezimal) den Ziffern 0 bis 9. Aus der Abfolge der Ziffern lässt sich der entsprechende Zahlenwert berechnen. Dabei gelten alle Zeichenwerte ausserhalb des Bereichs 48 bis 57 als Ende der Ziffernfolge.

Beispiel: Die Folge 49 52 50 32 entspricht der Zahl $1 \cdot 100 + 4 \cdot 10 + 2 \cdot 1 = 142$, der Wert 32 zeigt also das Ende der Ziffernfolge an.

Es wird nun ein 80386-Unterprogramm `atoi` benötigt, das die gegebene Bytefolge in den entsprechenden Zahlenwert umwandelt.

Zusätzliche Anforderungen:

- Der Start der Bytefolge wird im Register `edi` übergeben, nach dem Aufruf soll `edi` auf den Zeichenwert zeigen, der die Ziffernfolge beendet hat.
- Das umgewandelte Ergebnis soll in Register `eax` stehen.
- Außer `edi` und `eax` sollen nach dem Aufruf keine Register verändert sein.

(e) In dieser Aufgabe soll die Eingabe des Taschenrechners ausgewertet werden. Zur Vereinfachung ist nur noch die Addition erlaubt, die durch den Zeichenwert 43 für '+' signalisiert wird. Die Trennung zwischen zwei Zahlenfolgen wird mit dem Zeichenwert 32 (Leerzeichen) angedeutet, das Ende der Eingabe mit dem Wert 46 für '.'. Neben diesen Werten und den 10 Werten für die Ziffern kommen keine weiteren in der Eingabefolge vor.

Beispiel: Die Zahlenfolge 49 50 32 49 32 50 32 43 32 43 32 46 entspricht der Eingabe „12 1 2 + + “. Die Berechnung soll also 15 ergeben.

Das folgende, kommentierte 80386-Programmgerüst kann solche Eingaben verarbeiten. Die Adresse des ersten Zeichens der Eingabe stehe in Register `edi`.

Ergänzen Sie die fehlenden Stellen mit den passenden Befehlen bzw. Befehlsparametern.

```
parse:
        cmp byte [___], ___     ; Ende der Eingabe?
        je __________           ; Ja

        cmp _________           ; Leerzeichen?
        je next_char            ; Ja

        cmp _________           ; Additionszeichen?
        jne _________           ; Nein, eine Ziffer

        call ________           ; Werte addieren

next_char:
        inc edi                 ; Ein Zeichen weiterschalten
        jmp parse               ; Nächstes Zeichen lesen

number:
        call _________          ; Ziffern in Zahl umwandeln
        _______________         ; Auf den Stack geben
        jmp parse

end_parse:
        ...
```

6. (Ähnlich einer alten Klausuraufgabe) Es soll ein Unterprogramm `set_pixel` geschrieben werden, das Pixel in einem Bildschirmspeicher setzt. Der Bildschirmspeicher hat die Auflösung 1024×768, ist monochrom und linear adressiert. Jedes Bit in einem Byte definiert, ob das Pixel gesetzt ist (1: gesetzt, 0: nicht gesetzt), dabei entspricht Bit 7 dem linken Pixel und Bit 0 dem rechten Pixel. Der Start des Bildschirmspeichers stehe an der Speicherstelle `display_start`. Die Koordinaten x und y werden in den Registern `eax` und `ebx` übergeben.

   (a) Welche Formeln benötigt man, um die Speicheradresse bzw. die Bitnummer des zu verändernden Bytes für das Setzen eines Pixels auszurechnen?

   (b) Schreiben Sie das Unterprogramm `set_pixel`, das das Pixel setzt. Denken Sie daran, dass der Aufrufer der Funktion durchaus Koordinaten außerhalb des darstellbaren Bereichs angeben kann!