Praktikumsprotokoll – Mikrorechentechnik I
Versuch „Von-Neumann-Simulator“

Fabian Kurz, Alexander Eder
Stephan Stiebitz, Phillip Burker

5. November 2004

Inhaltsverzeichnis

1 Aufgabenstellung

Im Praktikumsversuch zum „Von-Neumann-Simulator“ ist ein Programm zu entwerfen, welches die die Operation

$$Y = X!$$

realisiert. Dazu wird eine ganzzahlige Ziffer $X : 0 \leq X \leq 4$ über die Tastatur eingelesen und deren Fakultät nach der Berechung auf dem Bildschirm ausgegeben.

2 Lösung

2.1 Programmentwurf, Algorithmus

Das Produkt der natürlichen Zahlen von 1 bis n bezeichnet man mit $n!$ („n–Fakultät). Zusätzlich ist festgelegt, daß $0! = 1$ ist.

Ein Algorithmus zur Berechnung der Fakultät läßt sich also verbal folgendermaßen beschreiben:

```
Lese ZAHL ein
Falls ZAHL gleich NULL ist, gib EINS aus und beende.
Sonst
{  Setze ERGEBNIS auf ZAHL
  Solange ZAHL nicht NULL ist
  { Multipliziere ERGEBNIS mit ZAHL, speichere in ERGEBNIS
    Dekrementiere ZAHL
  }
}
Gib ERGEBNIS aus
```

Da der im Praktikum verwendete Simulator jedoch als arithmetische Operationen lediglich Addition und Subtraktion beherrscht, muß die Multiplikation zur Bildung der Fakultät als wiederholte Addition ausgeführt werden. Hierfür ist eine weitere Schleife erforderlich, es ergibt sich folgender Algorithmus:

```
Lese FAKTOR1 und FAKTOR2 ein
Solange FAKTOR2 größer als 1 ist
 { Addiere FAKTOR1 zu SUMME
   Dekrementiere FAKTOR2
 }
Gib SUMME aus
```

Anhand des somit kompletten Algorithmus konnte das Programm einfach in Assemblersprache übertragen werden. Der komplette und ausführlich dokumentierte Quelltext befindet sich im Anhang A.

2.2 Probleme

Nennenswerte Probleme traten weder bei der Findung des Algorithmus, noch bei der Implementierung in Assemblersprache auf. Die erste lauffähige Version lieferte ein Ergebnis von $n! \cdot (n-1)$, was jedoch mithilfe des Debuggers schnell darauf zurückzuführen war, daß die Zwischensumme der Multiplikation nicht korrekt zurückgesetzt wurde. Durch das Setzen geeigneter Breakpoints

konnte der Debug–Vorgang deutlich beschleunigt werden, da somit das Programm nicht von Anfang an im Einzelschrittbetrieb durchlaufen mußte, sondern erst an der Stelle, an der der Fehler vermutet wurde vom schnellen Betrieb in den Einzelschrittberieb gewechselt werden konnte.

Das fertige Programm liefert nun für Eingabewerte von 0 bis 4 korrekt den Wert der Fakultät der Zahl. Auf eine Überprüfung der Gültigkeit des Eingabewertes wurde verzichtet, da dies auch in der Aufgabenstellung nicht verlangt wurde und den Quelltext unnötig unübersichtlich gemacht hätte.

Die Schleife zur Multiplikation ist so gewählt, daß möglichst wenige Durchläufe benötigt werden. So wird z.B. $2 \cdot 6$ als $6 + 6$ und nicht als $2 + 2 + 2 + 2 + 2 + 2$ berechnet.

3 Genaue Betrachtung eines Programmabschnitts

Es soll dargestellt werden, wie die Multiplikation zweier Zahlen funktioniert. In diesem Beispiel werden die Zahlen 3 und 4 multipliziert. Die 4 wird dreimal mit sich selbst addiert, nach der dritten Addition ist die Bedingung für den Rücksprung nicht mehr erfüllt, somit ist die Multiplikation komplett.

BZ_{alt}	BR_{alt}	BZ_{neu}	AC_{bin}	$Flags_{bin}$	Bemerkungen
09	LDA 34	A	11	0010	Laden des ersten Faktors
A	STA 35	B	11	0010	Sicherung des Faktors als Index
B	LDA 35	C	00	0010	Laden von Null
C	STA 36	D	00	0010	Summe auf Null setzen
D	LDA 36	E	00	0010	Summe wieder Laden
E	ADD 37	F	100	0010	Index zu Summe addieren
F	STA 36	10	100	0010	Summe sichern
10	LDA 35	11	11	0010	Index laden
11	SUB 31	12	10	0010	Index dekrementieren
12	STA 35	13	10	0010	Index sichern
13	CMP 33	14	10	0010	Prüfe Rücksprungbedingung
14	JGT 13	D	10	0010	Erfüllt, erneute Addition!
D	LDA 36	E	100	0010	Lade Summe
E	ADD 37	F	1000	0010	Index zu Summe addieren
F	STA 36	10	1000	0010	Summe sichern
10	LDA 35	11	10	0010	Index laden
11	SUB 31	12	1	0010	Index dekrementieren
12	STA 35	13	1	0010	Index sichern
13	CMP 33	14	1	0010	Prüfe Rücksprungbedingung
14	JGT 13	D	10	0010	Erfüllt, erneute Addition!
D	LDA 36	E	1000	0010	Lade Summe
E	ADD 37	F	1100	0010	Index zu Summe addieren
F	STA 36	10	1100	0010	Summe sichern
10	LDA 35	11	1	0010	Index laden
11	SUB 31	12	0	0100	Index dekrementieren
12	STA 35	13	0	0100	Index sichern
13	CMP 33	14	0	0100	Prüfe Rücksprungbedingung
14	JGT 13	15	0	0100	Nicht erfüllt! Fertig!
15	LDA 36	16	1100	0100	Lade Produkt in ACU

A Quelltext

```
; Mikrorechentechnik 1-Praktikum, Gruppe 63
;
; Versuch "Von-Neumann-Simulator"

; ###############  Initialisierung, Einlesen der Zahl, Umwandlung ASCII->Dez

       EXT  UPAUS   ; Externes Label UPAUS wird vereinbart
       RDA          ; lese Zahl (0 .. 4) deren Fakultaet gebildet werden soll
       SUB  ANULL   ; aus ASCII-Wert Dezimalzahl machen

; ###############  Test auf Sonderfall (0! := 1)

       CMP  NULL    ; ueberpruefe, ob Zahl Null ist
       JEQ  ZERO    ; Falls Zahl = 0, springe zu ZERO (Sonderfall, 0! = 1)

; ###############  Hier Fakultaet berechnen

       STA  FAK     ; Eingegebene Zahl in FAK sichern
LOOP   CMP  EINS    ; Akku (Zahl oder Laufindex) wird mit 1 vergleichen
       JEQ  FERTIG  ; Falls ACU=1 fertig -> Springe zum Label FERTIG
       SUB  EINS    ; (sonst) Index erniedrigen
       STA  INDEX   ; als Index sichern

; ###############  Hier Multiplikation von FAK mit INDEX

       LDA  INDEX   ; Laufindex der Fakultaet (zweiter Faktor) laden (obsolet)
       STA  INDEXM  ; in INDEXM sichern, da er veraendert wird
       LDA  NULL    ; ACU auf Null setzen
       STA  SUMME   ; Summe auf Null setzen
AGAIN  LDA  SUMME   ; Zwischensumme der Multiplikation laden
       ADD  FAK     ; Fakultaets-Zwischenwert aufsummieren  (FAK=erster Faktor)
       STA  SUMME   ; Zwischensumme sichern
       LDA  INDEXM  ; ehem. Laufindex der Fakultaet laden (zweiter Faktor)
       SUB  EINS    ; Laufindex dekrementieren, d.h. einmal weniger aufaddieren
       STA  INDEXM  ; neuen Index abspeichern
       CMP  NULL    ; wenn Laufindex 0, muss nicht weiter addiert werden
       JGT  AGAIN   ; wenn Laufindex > 0, muss weiter addiert werden
       LDA  SUMME   ; Fertig, SUMME in ACU laden

; ###############  Hier endet die Multiplikation

       STA  FAK     ; Ergebnis der Multiplikation sichern
       LDA  INDEX   ; Laufindex in ACU
       JMP  LOOP    ; wieder zurueck zum Anfang der Fakultaetsbildung

FERTIG LDA  FAK     ; Fakultaet fertig, Ergebnis FAK in ACU

; ###############  Hier die Fakultaet fertig berechnet

       JMP  GIBAUS  ; zur Ausgabe springen
```

```
; ###############  Behandlung des Sonderfalles 0!

ZERO   LDA  EINS    ; Bei 0! wird als Ergebnis 1 in den ACU geladen

; ###############  Ausgabe auf den Bildschirm, Beenden des Programmes

GIBAUS JSB  UPAUS   ; Ausgabe des ACU auf den Bildschirm durch UPAUS
       HLT          ; Programmausfuehrung wird beendent

; ###############  Initialisierungen

ANULL  DEC 48       ; ASCII Wert fuer die 0
EINS   DEC 1        ; Wert 1
NULL   DEC 0        ; Wert 0
INDEX  DEC 0        ; Laufindex fuer Fakultaet
INDEXM DEC 0        ; Laufindex fuer Multiplikation
SUMME  DEC 0        ; Zwischensumme bei Multiplikation
FAK    DEC 0        ; Zwischenprodukt der Fakultaet
```