MIMA
Die Sicht von außen

Aufgabensammlung & Lösungen
zur MIMA-Assemblersprache

1. Juni 2004

Vorwort

Weniger ist bekanntlich mehr. Auf diesem Prinzip baut die Architektur der MIMA auf. Das Befehlsformat der MIMA hat theoretisch Platz für 31 unterschiedliche Befehle, wir werden jedoch mit einer Auswahl von 17 Befehlen in der gesamten Aufgabensammlung auskommen. Zwar wünscht man sich einen **SUB a** Befehl, aber vielleicht findet jemand einen wichtigeren Befehl, der dann den OpCode E bekommen soll. Wer einen Befehl vorschlägt, sollte unbedingt das Mikroprogramm dazu entwickeln können. Sehr interessant wäre ein **MUL a** Befehl. Dieses ist mir jedoch nicht in der Mikroprogrammierung gelungen. Vielleicht habt ihr da mehr Glück? Immerhin existiert eine MIMA Implementierung der Addiere und Schiebemethode, so dass zumindest als Unterprogramm eine Multiplikation durchgeführt werden kann.

Die MIMA kennt 2 Befehlsformate: Das erste Format wird für die Übergabe von Adressen oder Konstanten benutzt. Diese können maximal 20 Bit breit sein. Der Arbeitsspeicher der MIMA ist aber wegen des 20 Bit breiten Adressbusses sowieso auf 1M Adressen begrenzt, also auf 3 MByte. Das zweite Format kann keine Parameter übergeben. Hier werden Befehle wie HALT oder Befehle die nur auf dem Akku operieren, z.B. rotieren und invertieren implementiert.

Von außen Zugreifbar ist für den Programmierer nur der 24 Bit breite Akku. Mehr gibt es für den MIMA Softwareentwickler nicht. Dies macht die MIMA sehr übersichtlich. Werden weitere Variablen benötigt, so sind diese im relativ großem Arbeitsspeicher abzulegen.

Blatt1 hat sich ausführlich mit der internen Funktion der MIMA auseinandergesetzt. Es wurden für das Steuerwerk alle Befehle als Mikroprogrammcode geschrieben. Es existiert ein Applet zur Visualisierung dieser internen Abläufe.

Was fehlt noch?

- Es fehlt ein Developerkid für den echten MIMA Softwareentwickler. Nachdem die Hardware ausreichend entwickelt und simuliert wurde, will auch gesehen werden, wie leistungsfähig der Befehlssatz in der Praxis ist.
- Es fehlen interessante Aufgabenstellungen.

Was will dieses Skript?

Dieses Skript liefert die gewünschten interessanten Aufgabenstellungen mit jeweils einem Lösungsvorschlag. Bekanntlich führen mehrere Wege zum Ziel. Die hier ange-

geben Lösungen wurden zwar mehrfach geprüft, dennoch können Fehler vorkommen. Ich bitte daher um Benachrichtigung, wenn sich ein Fehler eingeschlichen haben sollte. Weitere Aufgabenstellungen oder Programme werden gerne aufgenommen. Falls du einen guten Simulator oder gar eine Entwicklungsumgebung für die MIMA gefunden oder geschrieben hast, so würde ich das auch gerne an dieser Stelle veröffentlichen. Ansonsten wünsche ich Euch viel Spaß mit den theoretisch funktionierenden Programmen :-)

Befehlsübersicht

Bevor es ans Eingemachte geht, hier noch eine Befehlsübersicht. Mehr Befehle sind nicht vorgesehen. Befehlswünsche bitte nur mit Mikroprogrammcode (siehe Vorwort) einreichen!

OpCode	**Mnemonik**	**Beschreibung**
0	LDC c	c → Akku
1	LDV a	<a> → Akku
2	STV a	Akku → <a>
3	ADD a	Akku + <a> → Akku
4	AND a	Akku AND <a> → Akku
5	OR a	Akku OR <a> → Akku
6	XOR a	Akku XOR <a> → Akku
7	EQL a	if (Akku==<a>) { -1 → Akku;} else { 0 → Akku;}
8	JMP a	Sprung zur Adresse a
9	JMN a	wenn Akku<0, dann Sprung nach a
A	LDIV a	<<a>> → Akku
B	STIV a	Akku → <<a>>
C	JMS a	jump subroutine (s.u.)
D	JIND a	jump indirect (s.u.)
E		frei
F0	HALT	stoppt die MIMA
F1	NOT	Einskomplement von Akku → Akku
F2	RAR	rotiere Akku eins nach rechts → Akku (s.u)
F3 - FF		frei

Anmerkungen:

- Was beim **RAR** Befehl rechts rausgeschoben wird, wird links wieder eingeführt.
- Der Befehl **JMS ziel** bewirkt, daß die Adresse des nachfolgenden Befehls, also die Rücksprungadresse, an der Speicheradresse ziel abgelegt und dann zum Befehl mit der Adresse ziel+1 verzweigt wird.
- Der Befehl **JIND ziel** bewirkt, dass zu derjenigen Adresse verzweigt wird, die in der Speicherzelle ziel abgespeichert ist.

Primzahltest

Es soll festgestellt werden, ob es sich bei der Zahl in Speicherzelle 0x00 um eine Primzahl handelt. Falls nicht, so soll als Beweis, die Zahl durch einen beliebigen Teiler der Zahl ersetzt werden. Primzahlen bleiben unverändert stehen. Nehmen Sie der Einfachheit wegen an, dass die Zahl in Zelle 0x00 zwischen 2 und 2^{23}-1 groß ist. Das Programm soll an Adresse 0x100 beginnen. (Lösung auf Seite 4)

Zahlenraten

Schreibe ein Computerspiel für die MIMA! Eine Person schreibt an Speicherzelle 0x01 eine beliebige Zahl. Der Spieler 2 muss versuchen diese Zahl zu erraten. Er gibt seinen Tipp in Speicherzelle 0x02 ein. Das MIMA Programm sagt nun, ob die geratene Zahl kleiner, größer oder gleich der vorgegebenen Zahl war gemäß folgender Tabelle:

Tipp	Zelle 0x03
kleiner	-1
größer	$+1$
gleich	0

Das Programm soll an Speicheradresse 0x100 beginnen und antwortet in Zelle 0x03. Die Zahlen sollen als 2er Komplementzahlen interpretiert werden. Es sollen nur positive Zahlen geraten und eingegeben werden können. Hält sich Spieler 1 nicht an die Vereinbarung, so gibt das Programm in Zelle 0x03 eine 2 aus und bricht sofort ab. (Einen Lösungsvorschlag finden Sie auf Seite 5)

Wochentagsberechnung

Es soll ein MIMA-Programm geschrieben werden, das den Wochentag zu einem beliebigen (TAG, MONAT) im Jahr 2004 berechnet. Dazu werden die Wochentage wie folgt definiert:

0 Sonntag
1 Montag
2 Dienstag
3 Mittwoch
4 Donnerstag
5 Freitag
6 Samstag

Das Programm soll an der Speicheradresse 0x00100 beginnen. Der TAG steht in der Speicherzelle 0x00020 zur Verfügung und der MONAT steht an Adresse 0x00021 bereit. Nach Ablauf des Programms soll sich die Wochentagszahl an Adresse 0x00030 befinden. Sie können sich bei der Berechnung an folgendem c-Code orientieren:

```
main()
{
  int ersterTag[]={-1,4,0,1,4,6,2,4,0,3,5,1,3};
  int TAG = 3;
  int MONAT = 6;
  int wochentag = TAG + ersterTag[MONAT] - 1;
```

```
  wochentag%=7;

  printf(‘‘Wochentag=%i‘‘,wochentag);
}
```

Die jeweiligen Wochentage des ersten Tages im Monat stehen hier im Array erster-Tag zur Verfügung. Addiert man zu diesem Wochentag den (TAG-1) und rechnet modulo 7, so erhält man den gewünschten Wochentag wie oben definiert. Beachten Sie hierbei, dass in c die Arrayelemente mit 0 beginnend nummeriert werden, während der Kalender immer bei 1 anfängt zu zählen. (Lösung auf Seite 6)

Addiere und Schiebemultiplikation

Schreiben Sie eine Implementierung der Addiere und Schiebemethode für die MIMA. Halten Sie sich dabei an den Algorithmus für Betrag und Vorzeichenzahlen aus der TI1 Vorlesung. (Skript Seite 182-184)

Der Speicher sei wie folgt belegt:
0x001 = M
0x002 = Q
0x003 = A
0x100 = Programmstart

Lösung auf Seite 7

Lösung: Primzahltest

```
; Ein und Ausgabezelle
0x00000  00BAFF  ZAHL:      DS 0xBAFF      ; Zahl die getestet werden soll
; Hilfsvariablen
0x00001  000000  ZAHL2:     DS 0
0x00002  000000  NULL:      DS 0           ; Konstante 0
0x00003  000001  EINS:      DS 1           ; Konstante 1
0x00004  FFFFFE  MINUS2:    DS 0xFFFFFE    ; Konstante -2
0x00005  7FFFFF  FILTER:    DS 0x7FFFFF    ; Filter 011..11b
0x00006  000000  TEST:      DS 0
0x00007  000000  TEST2:     DS 0           ; 2erKomplement von TEST
; Programmstart
0x00100  100000  START:     LDV ZAHL       ; Prüfe zwischen 2 und
0x00101  F20000              RAR            ; sqrt(ZAHL) < Zahl/2
0x00102  900104              JMN WEITER     ; Wenn Zahl gerade
0x00103  80011C              JMP DURCH2     ; dann durch 2 teilbar!
0x00104  500003  WEITER:    OR EINS        ; Zahl/2 auf ungerade Zahl runden
0x00105  400005              AND FILTER     ; Filtern des RAR durschubes
; Schleife: Prüfe sämtliche Teiler auf Erfolg
0x00106  300004  SCHLEIFE:  ADD MINUS2     ; TEST um 2 reduzieren
0x00107  200006              STV TEST       ; und wieder speichern
```

```
0x00108  900120              JMN ENDE        ; | TEST=negativ oder
0x00109  700003              EQL EINS        ; | TEST=1
0x0010A  900120              JMN ENDE        ; | => Zahl war Primzahl!
0x0010B  C0010E              JMS PZTEST      ; Prüfe die Teilbarkeit
0x0010C  100006              LDV TEST
0x0010D  800106              JMP SCHLEIFE
; Unterprogramm: Teilbarkeit durch TEST prüfen
0x0010E  000000  PZTEST:     DS 0            ; Rücksprungadresse
0x0010F  100006              LDV TEST
0x00110  F10000              NOT             ; Berechne das 2er Komplement
0x00111  300003              ADD EINS        ; von TEST für
0x00112  200007              STV TEST2       ; fortlaufende Subtraktion
0x00113  100000              LDV ZAHL
; Forlaufende Subtraktion
0x00114  300007  SCHLEIFE2:  ADD TEST2       ; ZAHL2-=TEST
0x00115  90011B              JMN RETURN      ; nicht teilbar
0x00116  200001              STV ZAHL2
0x00117  700002              EQL NULL        ; | teilbar, denn TEST passt
0x00118  90011E              JMN KEINEPZ     ; | genau in ZAHL
0x00119  100001              LDV ZAHL2       ; EQL hat Akku zerstört
0x0011A  800114              JMP SCHLEIFE2
0x0011B  D0010E  RETURN:     JIND PZTEST     ; Return
; Ende:
0x0011C  000002  DURCH2:     LDC 2           ; Zahl durch 2 teilbar
0x0011D  80011F              JMP STORE
0x0011E  100006  KEINEPZ:    LDV TEST        ; Zahl durch TEST teilbar
0x0011F  200000  STORE:      STV ZAHL        ; Ergebnis speichern
0x00120  F00000  ENDE:       HALT
```

Lösung: Zahlenraten

```
; Ein und Ausgabe
0x00001  000071  ZAHL:       DS 113          ; Zahl von Spieler1
0x00002  0000F4  TIPP:       DS 244          ; Tipp von Spieler2
0x00003  000000  AUSGABE:    LDC 0           ; Bewertung des Tipps
; Hilfsvariablen
0x00004  FFFFFF  MINUS1:     DS 0xFFFFFF     ; Konstante -1
0x00005  000000  NULL:       DS 0            ; Konstante 0
0x00006  000001  EINS:       DS 1            ; Konstante 1
; Beginn des Programmes
0x00100  100001  START:      LDV ZAHL
0x00101  90010E              JMN ERROR       ; Zahl negativ?
0x00102  F10000              NOT             ; | Berechne die Differenz
0x00103  300006              ADD EINS        ; | zwischen den
0x00104  300002              ADD TIPP        ; | Zahlen
0x00105  90010C              JMN MINUS       ; Zahl zu klein?
0x00106  700005              EQL NULL        ; Zahl geraten?
0x00107  90010A              JMN TREFFER
0x00108  000001              LDC 1           ; Dann zu groß! :)
```

```
0x00109  80010F            JMP ENDING
0x0010A  000000  TREFFER:  LDC 0          ; Herzlichen Glückwunsch:
0x0010B  80010F            JMP ENDING     ; Zahl wurde erraten!
0x0010C  100004  MINUS:    LDV MINUS1     ; Spieler 1 hat
0x0010D  80010F            JMP ENDING     ; zu klein geraten
0x0010E  000002  ERROR:    LDC 2          ; Fehlerzahl!
0x0010F  200003  ENDING:   STV AUSGABE    ; Antwort ausgeben
0x00110  F00000            HALT
```

Lösung: Wochentagsberechnung

```
; Definition der ersten Tage im Monat:
0x00001  000004  ARRAY:    DS 4           ; Januar (Do)
0x00002  000000            DS 0           ; Februar (So)
0x00003  000001            DS 1           ; März (Mo)
0x00004  000004            DS 4           ; April (Do)
0x00005  000006            DS 6           ; Mai (Sa)
0x00006  000002            DS 2           ; Juni (Di)
0x00007  000004            DS 4           ; Juli (Do)
0x00008  000000            DS 0           ; August (So)
0x00009  000003            DS 3           ; September (Mi)
0x0000A  000005            DS 5           ; Oktober (Fr)
0x0000B  000001            DS 1           ; November (Mo)
0x0000C  000003            DS 3           ; Dezember (Mi)
; Eingabe:
0x00020  000003  TAG:      DS 3           ; Tag
0x00021  000006  MONAT:    DS 6           ; Monat
; Hilfsvariablen
0x00022  FFFFF9  MINUS7:   DS 0xFFFFF9    ; -7
0x00023  FFFFFF  MINUS1:   DS 0xFFFFFF    ; -1
0x00024  000007  PLUS7:    DS 7           ; 7
; Ausgabe
0x00030  000000  ERG:      DS 0           ; Ergebniszelle
; Hauptprogramm
0x00100  100021  START:    LDV MONAT      ; Akku=Monat
0x00101  300103            ADD BEFEHL     ; OpCode für LDV dazu addieren
0x00102  200103            STV BEFEHL     ; Befehl überschreiben
0x00103  100000  BEFEHL:   LDV 0          ; Akku=ersterTagimMonat[Monat]
0x00104  300020            ADD TAG        ; Akku+=Tag
0x00105  300023            ADD MINUS1     ; Akku-=1;
0x00106  300022  SCHLEIFE: ADD MINUS7     ; | Be-
0x00107  900109            JMN RAUS       ; | rechnung
0x00108  800106            JMP SCHLEIFE   ; | von
0x00109  300024  RAUS:     ADD PLUS7      ; | Akku%=7
0x0010A  200030            STV ERG        ; Wochentag in 0x30 schreiben
0x0010B  F00000  ENDE:     HALT
```

Lösung: Addiere und Schiebemultiplikation

```
; Hilfsvariablen
0x00001 | 000069 | M:          DS 0x69
0x00002 | 0000C6 | Q:          DS 0xC6
0x00003 | 000000 | A:          DS 0
0x00004 | 000000 | A2:         DS 0
0x00005 | 00007F | SCOUNT:     DS 0x7F        ; Schleifenzähler auf binärer Ebene
0x00006 | 00007F | VZDEL:      DS 0x7F        ; Vorzeichen Löschmaske 01111111b
0x00007 | 000080 | VZSET:      DS 0x80        ; Vorzeichen Setzmaske 10000000b
0x00008 | 000000 | VZ:         DS 0           ; Vorzeichen von M*Q = V(0)
0x00009 | 0000FF | FILTERA:    DS 0xFF        ; finaler Filter für A 11111111b
0x0000A | 0000FE | FILTERQ:    DS 0xFE        ; finaler Filter für Q 11111110b
; Programmstart:
0x00100 | 100001 | START:      LDV M          ; Vorzeichen von M*Q
0x00101 | 600002 |             XOR Q          ; an die Stelle VZ
0x00102 | 200008 |             STV VZ         ; speichern
0x00103 | 100005 | SCHLEIFE:   LDV SCOUNT     ; Schleifenzähler laden
0x00104 | F20000 |             RAR            ; und schieben und wieder
0x00105 | 200005 |             STV SCOUNT     ; speichern
0x00106 | 900108 |             JMN NEXT       ; Z<7? Ja:
0x00107 | 80011F |             JMP NACH7      ; Nein
0x00108 | 100002 | NEXT:       LDV Q          ; Ist
0x00109 | F20000 |             RAR            ; Q(7)==1?
0x0010A | 90010C |             JMN ADD        ; Ja:
0x0010B | 800113 |             JMP SHIFT      ; Nein
; Addiere A+M und schreibe in A:
0x0010C | 100003 | ADD:        LDV A          ; Vorzeichen
0x0010D | 400006 |             AND VZDEL      ; von A
0x0010E | 200004 |             STV A2         ; filtern
0x0010F | 100001 |             LDV M          ; Vorzeichen von M
0x00110 | 400006 |             AND VZDEL      ; filtern und
0x00111 | 300004 |             ADD A2         ; M+A berechnen und
0x00112 | 200003 |             STV A          ; in A speichern
; Schiebe:
0x00113 | 100002 | SHIFT:      LDV Q          ; Q laden und
0x00114 | F20000 |             RAR            ; nach rechts schieben
0x00115 | 200002 |             STV Q          ; und speichern
0x00116 | 100003 |             LDV A          ; A laden und
0x00117 | F20000 |             RAR            ; nach rechts schieben
0x00118 | 200003 |             STV A          ; und speichern.
0x00119 | 90011B |             JMN SET1       ; war A(7) eine 1? Ja:
0x0011A | 800103 |             JMP SCHLEIFE   ; Nein
0x0011B | 100002 | SET1:       LDV Q          ; | Q laden und bit Nr.
0x0011C | 500007 |             OR VZSET       ; | 0 setzen und wieder
0x0011D | 200002 |             STV Q          ; | speichern
0x0011E | 800103 |             JMP SCHLEIFE
; Nach 7 Schleifendurchgängen:
```

```
0x0011F  100008  NACH7:  LDV VZ        ; Vorzeichen laden
0x00120  400007          AND VZSET     ; und filtern
0x00121  700007          EQL VZSET     ; Vorzeichen==1?
0x00122  900124          JMN DOSET     ; Ja:
0x00123  800127          JMP NOSET     ; Nein
0x00124  100003  DOSET:  LDV A         ; | A(0)
0x00125  500007          OR VZSET      ; | auf 1
0x00126  200003          STV A         ; | setzen
0x00127  100003  NOSET:  LDV A         ; A
0x00128  400009          AND FILTERA   ; säu-
0x00129  200003          STV A         ; bern
0x0012A  100002          LDV Q         ; Q
0x0012B  40000A          AND FILTERQ   ; säu-
0x0012C  200002          STV Q         ; bern
0x0012D  F00000          HALT
```