Freie Universität Berlin

Technische Informatik II
Rechnerarchitektur

2.Lade- & Speicherbefehle in MMIX (Teil 1)

Matthias Dräger

E-Mail: mdraeger@mi.fu-berlin.de
www: www.matthias-draeger.info/lehre/sose2010ti2/
tinyurl.com/sose2010ti2

Zuletzt bearbeitet: 26.04.2010

Wiederholung

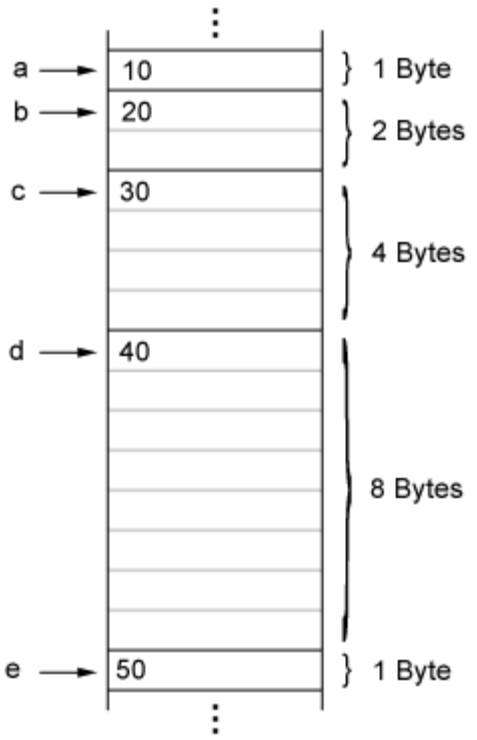
- Pseudobefehle zum reservieren von Speicherplatz:
  - BYTE
    - pro Datensatz (eine Zahl oder ein Zeichen) wird ein Byte reserviert
    - Zahlen müssen im Bereich 0 bis 255 liegen
  - WYDE
    - pro Datensatz werden zwei Bytes reserviert
    - Zahlen müssen im Bereich 0 bis 65.535 liegen
  - TETRA
    - pro Datensatz werden vier Bytes reserviert
    - Zahlen müssen im Bereich 0 bis $2^{32}-1$ = 4.294.967.295 liegen
  - OCTA
    - pro Datensatz werden acht Bytes reserviert
    - Zahlen müssen im Bereich 0 bis $2^{64}-1$ = 18.446.744.073.709.551.615 liegen

Laden vom Daten-Segment

Vergleich der Pseudobefehle

Wie sind Daten im Hauptspeicher angeordnet?

```
% ladebefehle.mms

            LOC         Data_Segment
            GREG @
% Verschiedene Datensätze anlegen
a           BYTE        10
b           WYDE        20
c           TETRA       30
d           OCTA        40
e           BYTE        50

            LOC #100
sum         IS $5       % Synonym für $4 anlegen

Main        LDB $0,a                % lade den Wert 10 in Register $0
            LDW $1,b                % lade den Wert 20 in Register $1
            LDT $2,c                % lade den Wert 30 in Register $2
            LDO $3,d                % lade den Wert 40 in Register $3
            LDB $4,e                % lade den Wert 50 in Register $4
            ADD sum,$0,$1           % sum = 10 + 20
            ADD sum,sum,$2          % sum = sum + 30
            ADD sum,sum,$3          % sum = sum + 40
            ADD sum,sum,$4          % sum = sum + 50
            TRAP 0,Halt,0           % beende Programm
```



Die Marken a bis e zeigen auf die Daten im Hauptspeicher.

Ladebefehle im Überblick

Name: LDB (Load Byte)

Spez.: LDB $X,*Marke*
LDB $X,$Y,*Offset*

Beschreibung:
Es wird ein Byte von der Stelle *Marke* oder von der Adresse $Y (Register) geladen und nach $X gespeichert.

Zahlenbereich: -128 bis 127

Name: LDT (Load Tetra)

Spez.: LDT $X,*Marke*
LDT $X,$Y,*Offset*

Beschreibung: *analog zu LDB*

Zahlenbereich: -2^{31} bis $2^{31}-1$

Hinweis: Der Offset sollte ein Vielfaches von 4 sein.

Name: LDW (Load Wyde)

Spez.: LDW $X,*Marke*
LDW $X,$Y,*Offset*

Beschreibung: *analog zu LDB*

Zahlenbereich: -2^{15} bis $2^{15}-1$

Hinweis: Der Offset sollte ein Vielfaches von 2 sein.

Name: LDO (Load Octa)

Spez.: LDO $X,*Marke*
LDO $X,$Y,*Offset*

Beschreibung: *analog zu LDB*

Zahlenbereich: -2^{63} bis $2^{63}-1$

Hinweis: Der Offset sollte ein Vielfaches von 8 sein.

Hinweis: Der Offset ist eine ganze Zahl (oder ein Register mit einer ganzen Zahl), der zur Adresse hinzuaddiert wird. Er ist optional und kann auch weggelassen werden.

Probleme bei Ladebefehle

Welcher Wert steht in Register $0?

```
% problem-LDB.mms

            LOC Data_Segment
            GREG @
a           BYTE 200

            LOC #100
Main        LDB $0,a              % lade den Wert 200(?) in Register $0
            TRAP 0,Halt,0         % beende Programm
```

→ Wert -56 steht in Register $0.

Da 200 größer als 127 ist, entsteht ein Überlauf. Nach 127 kommen die Zahlen -128, -127, -126,…

Es wird also wie folgt gerechnet: $200 - 2^8 = -56$.

Bei WYDE, TETRA und OCTA geschieht dies analog:

- WYDE: *Zahl* – 2^{16}
- TETRA: *Zahl* – 2^{32}
- OCTA: *Zahl* – 2^{64}

Ladebefehle ohne Berücksichtigung des Vorzeichens

Name: LDBU (Load Byte unsigned)

Spez.: LDBU $X,*Marke*
LDBU $X,$Y,*Offset*

Beschreibung:
Es wird ein Byte von der Stelle *Marke* oder von der Adresse $Y (Register) **ohne Berücksichtigung des Vorzeichens** geladen und nach $X gespeichert.

Zahlenbereich: 0 bis 255

Name: LDTU (Load Tetra unsigned)

Spez.: LDTU $X,*Marke*
LDTU $X,$Y,*Offset*

Beschreibung: *analog zu LDBU*

Zahlenbereich: 0 bis 2^{32}-1

Hinweis: Der Offset sollte ein Vielfaches von 4 sein.

Name: LDWU (Load Wyde unsigned)

Spez.: LDWU $X,*Marke*
LDWU $X,$Y,*Offset*

Beschreibung: *analog zu LDBU*

Zahlenbereich: 0 bis 2^{16}-1

Hinweis: Der Offset sollte ein Vielfaches von 2 sein.

Name: LDOU (Load Octa unsigned)

Spez.: LDOU $X,*Marke*
LDOU $X,$Y,*Offset*

Beschreibung: *analog zu LDBU*

Zahlenbereich: 0 bis 2^{64}-1

Hinweis: Der Offset sollte ein Vielfaches von 8 sein.

Hinweis: Der Offset ist eine ganze Zahl (oder ein Register mit einer ganzen Zahl), der zur Adresse hinzu addiert wird. Er ist optional und kann auch weggelassen werden.

Zugriff auf eine Adresse

Name: LDA (Load Address)

Spez.: LDA $X,*Marke*

Beschreibung:
Es wird die Adresse von *Marke* bestimmt und in Register $X gespeichert. LDA nimmt Bezug auf eine Adresse in einem globalen Register, weshalb es notwendig ist GREG @ nach einem Sprung ins Daten-Segment anzuwenden.

Name: GETA (Get Address)

Spez.: GETA $X,*Marke*

Beschreibung:
Es wird die Adresse von *Marke* relativ bestimmt und in Register $X gespeichert. GETA kann nicht auf Marken zugreifen, die weiter als #FFFF Tetrabyte entfernt sind (wie z.B. das Daten-Segement).

Beispiel:

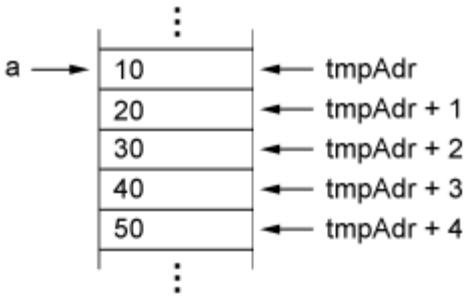
```
% beispiel-LDA.mms

            LOC Data_Segment
            GREG @
a           BYTE 50

            LOC #100
Main        LDA $0,a
            TRAP 0,Halt,0
```

→ In Register $0 steht die Adresse, wo die Zahl 50 im Speicher steht.

Mehrere Byte-Zahlen laden

Wie legt man eine Liste von Zahlen an?

```
% byte-zahlen.mms

            LOC Data_Segment
            GREG @
% einen Datensatz anlegen
a           BYTE 10,20,30,40,50

            LOC #100
sum         IS $5       % Synonym für $4 anlegen
tmpAdr      IS $6       % Register zum Speichern einer Adresse

Main        LDA tmpAdr,a     % lade die Adresse auf die die Marke a zeigt
            LDB $0,tmpAdr    % lädt den Wert von der Adresse tmpAddr -> 10 in Register $0
            LDB $1,tmpAdr,1  % tmpAddr+1 -> 20 in Register $1
            LDB $2,tmpAdr,2  % tmpAddr+2 -> 30 in Register $2
            LDB $3,tmpAdr,3  % tmpAddr+3 -> 40 in Register $3
            LDB $4,tmpAdr,4  % tmpAddr+4 -> 50 in Register $4

            ADD sum,$0,$1
            ADD sum,sum,$2
            ADD sum,sum,$3
            ADD sum,sum,$4
            TRAP 0,Halt,0 % beende Programm
```



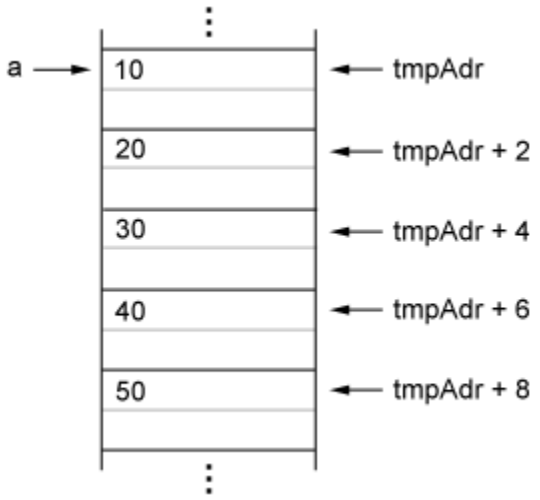
Welchen Wert hat *sum* beim Beenden des Programms?

→ 10 + 20 + 30 + 40 + 50 = 150

Mehrere Wyde-Zahlen laden

Gleiches Beispiel, nur mit WYDE:

```
% wyde-zahlen.mms

            LOC Data_Segment
            GREG @
% einen Datensatz anlegen
a           WYDE 10,20,30,40,50

            LOC #100
sum         IS $5       % Synonym für $4 anlegen
tmpAdr      IS $6       % Register zum Speichern einer Adresse

Main        LDA tmpAdr,a     % lade Adresse auf die die Marke a zeigt
            LDW $0,tmpAdr    % lädt Wert von Adresse tmpAddr
            LDW $1,tmpAdr,2
            LDW $2,tmpAdr,4
            LDW $3,tmpAdr,6
            LDW $4,tmpAdr,8

            ADD sum,$0,$1          % sum = 10 + 20
            ADD sum,sum,$2         % sum = sum + 30
            ADD sum,sum,$3         % sum = sum + 40
            ADD sum,sum,$4         % sum = sum + 50
            TRAP 0,Halt,0          % beende Programm
```



Warum muss man die Adresse *tmpAdr* immer um 2 erhöhen?

→ Weil die Daten (Zahlen) immer 2 Bytes von einander entfernt sind.

Speichern im Daten-Segment

Speicherbefehle im Überblick

Name: STB (Store Byte)

Spez.: STB $X,*Marke*
STB $X,$Y,*Offset*

Beschreibung:
Es wird der Wert aus dem Register $X an die Adresse der Stelle *Marke* gespeichert oder an die Adresse im Register $Y.

Zahlenbereich: -128 bis 127

Name: STT (Store Tetra)

Spez.: STT $X,*Marke*
STT $X,$Y,*Offset*

Beschreibung: *analog zu STB*

Zahlenbereich: -2^{31} bis $2^{31}-1$

Hinweis: Der Offset sollte ein Vielfaches von 4 sein.

Name: STW (Store Wyde)

Spez.: STW $X,*Marke*
STW $X,$Y,*Offset*

Beschreibung: *analog zu STB*

Zahlenbereich: -2^{15} bis $2^{15}-1$

Hinweis: Der Offset sollte ein Vielfaches von 2 sein.

Name: STO (Store Octa)

Spez.: STO $X,*Marke*
STO $X,$Y,*Offset*

Beschreibung: *analog zu STB*

Zahlenbereich: -2^{63} bis $2^{63}-1$

Hinweis: Der Offset sollte ein Vielfaches von 8 sein.

Hinweis: Der Offset ist eine ganze Zahl (oder ein Register mit einer ganzen Zahl), der zur Adresse hinzuaddiert wird. Er ist optional und kann auch weggelassen werden.

Achtung: Werden Zahlen außerhalb des Zahlenbereichs gespeichert, entsteht ein Überlauf, d.h., die Zahl wird „abgeschnitten". **Ein Überlauf wird im Spezialregister rA angezeigt.**

Speicherbefehle ohne Berücksichtigung des Vorzeichens

Name: STBU (Store Byte unsigned)

Spez.: STBU $X,*Marke*
STBU $X,$Y,*Offset*

Beschreibung:
Es wird der Wert aus dem Register $X an die Adresse der Stelle *Marke* gespeichert oder an die Adresse im Register $Y.

Zahlenbereich: 0 bis 255

Name: STTU (Store Tetra unsigned)

Spez.: STTU $X,*Marke*
STTU $X,$Y,*Offset*

Beschreibung: *analog zu STBU*

Zahlenbereich: 0 bis 2^{32}-1

Hinweis: Der Offset sollte ein Vielfaches von 4 sein.

Name: STWU (Store Wyde unsigned)

Spez.: STWU $X,*Marke*
STWU $X,$Y,*Offset*

Beschreibung: *analog zu STBU*

Zahlenbereich: 0 bis 2^{16}-1

Hinweis: Der Offset sollte ein Vielfaches von 2 sein.

Name: STOU (Store Octa unsigned)

Spez.: STOU $X,*Marke*
STOU $X,$Y,*Offset*

Beschreibung: *analog zu STBU*

Zahlenbereich: 0 bis 2^{64}-1

Hinweis: Der Offset sollte ein Vielfaches von 8 sein.

Hinweis: Der Offset ist eine ganze Zahl (oder ein Register mit einer ganzen Zahl), der zur Adresse hinzuaddiert wird. Er ist optional und kann auch weggelassen werden.

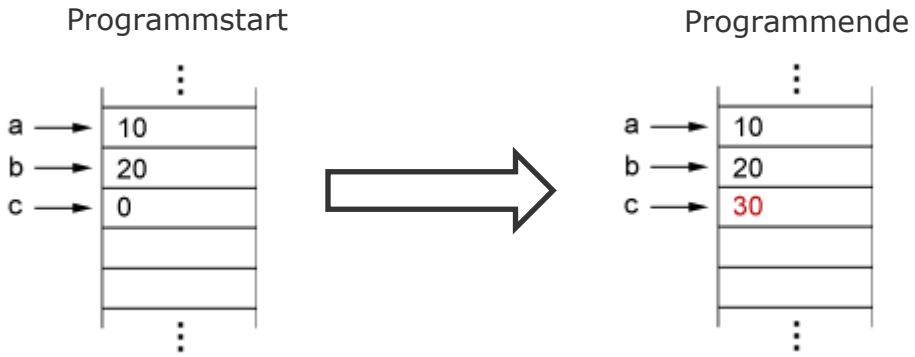
Achtung: Werden Zahlen außerhalb des Zahlenbereichs gespeichert, entsteht ein **Überlauf**, allerdings wird dieser **nicht signalisiert**!

Eine Byte-Zahl speichern

Ziel: Lade a und b, addiere die Zahlen und speichere das Ergebnis in c.

```
% beispiel-STB.mms

            LOC Data_Segment
            GREG @

a           BYTE 10
b           BYTE 20
c           BYTE 0

            LOC #100
Main        LDB $0,a              % lade 10 in $0
            LDB $1,b              % lade 20 in $1
            ADD $0,$0,$1          % a = a + b
            STB $0,c              % speichere das Ergebnis nach c
            TRAP 0,Halt,0
```

Was ändert sich im Hauptspeicher?



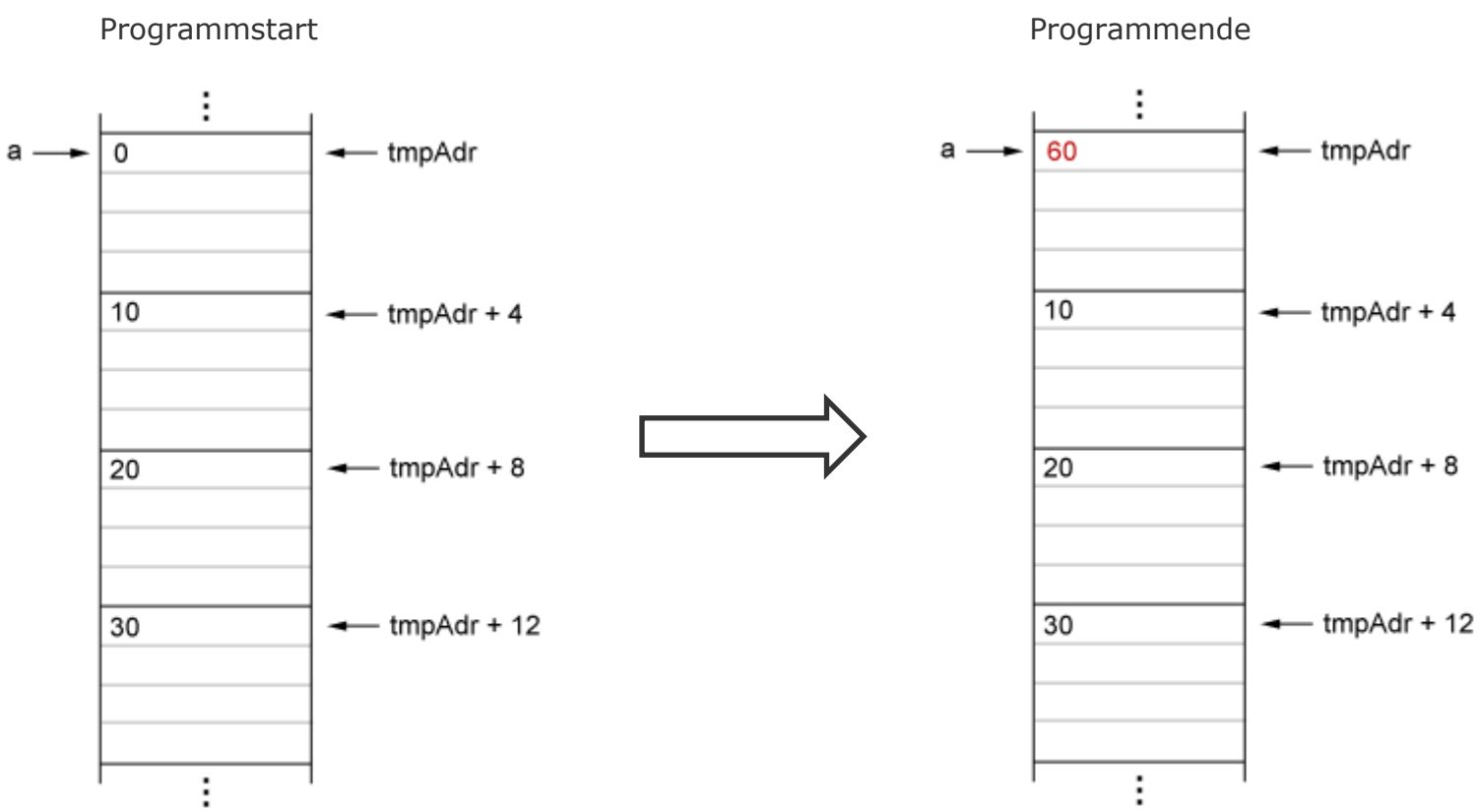
Tetra-Zahlen laden und speichern

Ziel: Addiere die Zahlen von 2. bis 4.Stelle (von der Marke *a*) und speichere das Ergebnis an die 1.Stelle.

```
% beispiel-STT.mms

            LOC Data_Segment
            GREG @
a           TETRA       0,10,20,30

            LOC #100
tmpAdr      IS          $0          % zum Zwischenspeichern der Lade-Adresse
no          IS          $1          % entählt die geladene Zahl
sum         IS          $2          % Summe aller geladener Zahlen

Main        LDA tmpAdr,a            % Adresse von a holen
            % wir überspringen die erste Zahl, denn da soll das Ergebnis hin
            LDT no,tmpAdr,4         % wir gehen 4 Byte weiter und laden 10 in no ($1)
            SET sum,no              % initialisiere die Summe mit der aktuellen Nummer
            LDT no,tmpAdr,8         % die 3.Zahl ist bereits 8 Byte entfernt -> lade 20 in no ($1)
            ADD sum,sum,no          % addiere die Zahl zur Summe
            LDT no,tmpAdr,12        % die letzte Zahl ist 12 Byte entfernt -> lade 30 in no ($1)
            ADD sum,sum,no          % addiere die Zahl zur Summe
            STT sum,a               % speichere das Ergebnis an die erste Stelle bei a

            % Alternative:
            STT sum,tmpAdr          % speichert das Ergebnis an die Adresse tmpAdr

            TRAP 0,Halt,0
```

Tetra-Zahlen laden und speichern (2)

Was ändert sich im Hauptspeicher?



Zusammenfassung

- Unterschied zw. BYTE, WYDE, TETRA, OCTA
- Zugriff auf Speicheradressen (LDA)
- Laden von Daten (mit Adressen oder Marken)
  - LDB, LDBU
  - LDW, LDWU
  - LDT, LDTU
  - LDO, LDOU
- Speichern von Daten (mit Adressen oder Marken)
  - STB, STBU
  - STW, STWU
  - STT, STTU
  - STO, STOU
- Benutzung von Offsets