



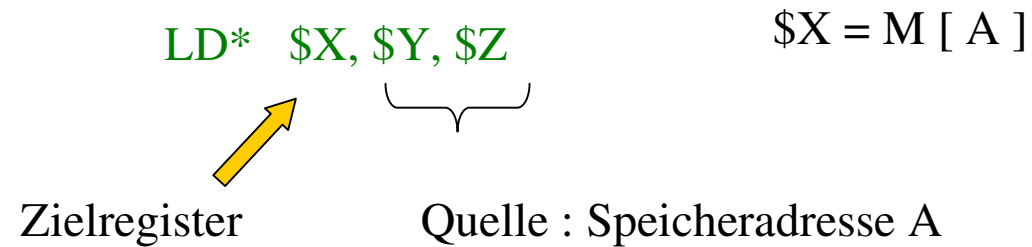
Adressierung



- Adressierung von Daten
bei LOAD/STORE – Befehlen
- Angabe der Zieladresse
bei Sprungbefehlen



LOAD / STORE - Befehle



* = O / T / W / B



Adressierung

Mögliche Speicheradresse **A**:

A = \$Y + Z LDOI/STOI immediate Operand Z

A = \$Y + \$Z LDO/STO Operand in Register \$Z

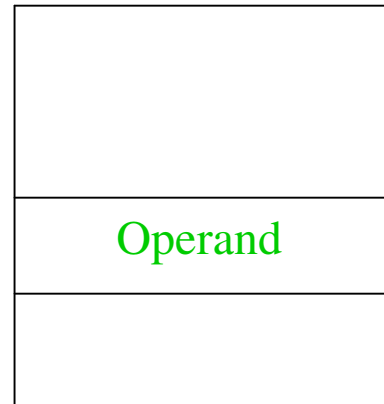
Erlaubte Werte für A  Alignment

Data_Segment

A



Operand





Registerindirekte Adressierung

$$Z = 0 \quad A = \$Y$$

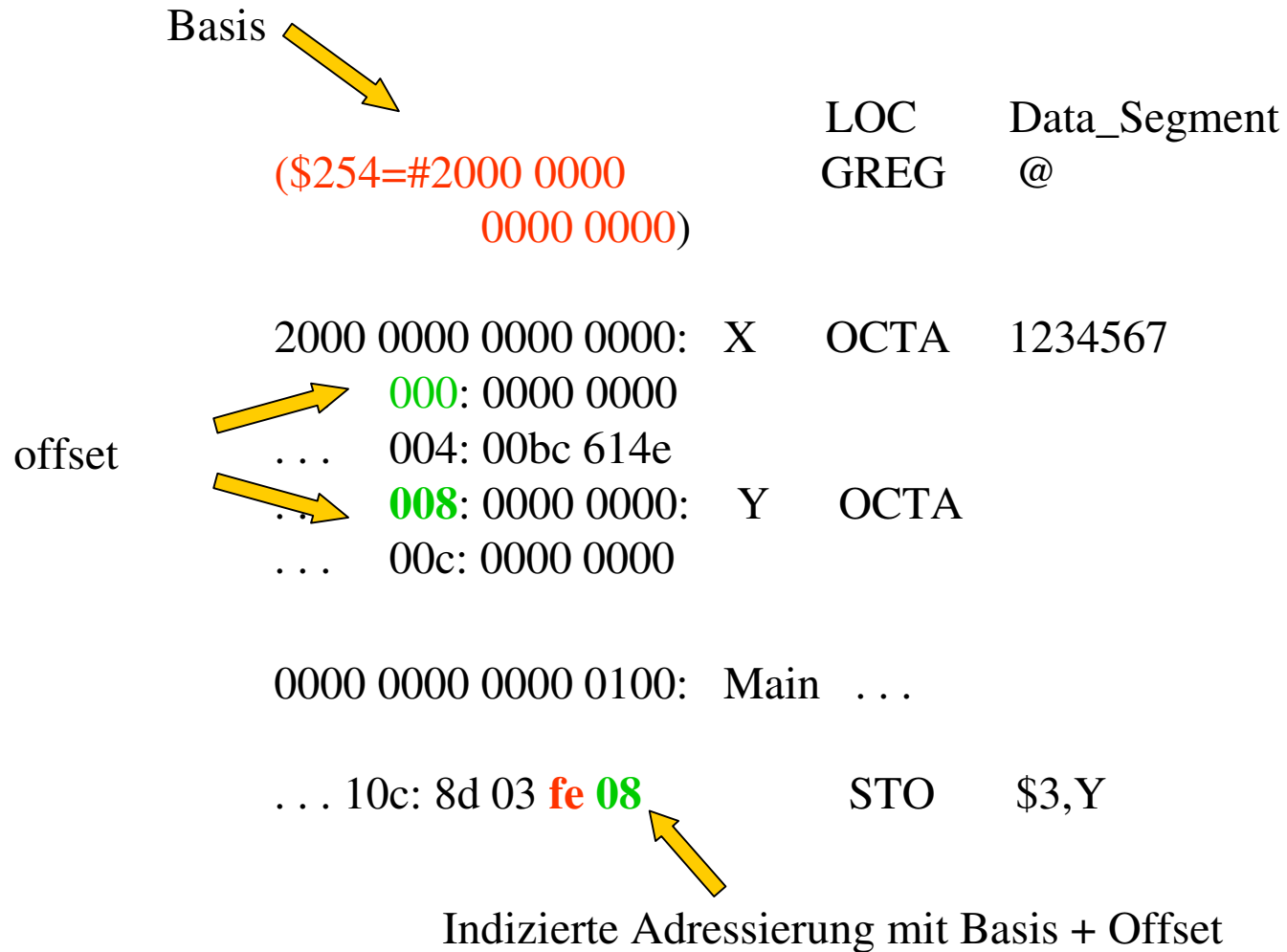
Absolute Adresse **A** in Register Y

Beispiel : Laden / Speichern einer Variablen

				LOC	Data_Segment
base	IS	\$1	X	OCTA	12345678
offset	IS	\$2	Y	OCTA	
				...	
			Main	LDO	\$3,X
				LDA	base,X
				LDO	\$3,base,0
				STO	\$3,Y



Adressierung





Adressierung

indizierte Adressierung

STO \$3,Y

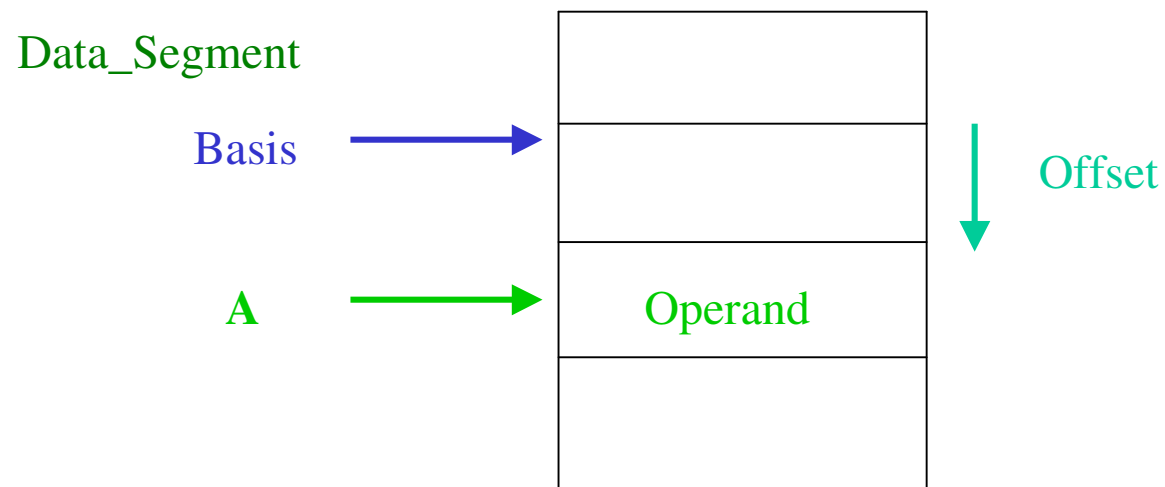
Basis in \$254

Offset = 08 (immediate)

... 10c: 8d 03 fe 08

$A = \text{Basis} + \text{Offset}$

Allgemein:





indizierte Adressierung

$$A = \text{Basis} + \text{Offset}$$

Basis in \$Y

Offset in \$Z

```
0000 0000 0000 0100: Main ...  
  
... 110: 23 01 fe 00      LDA  base,Data_Segment  
  
... 114: e3 02 00 08      SET  offset,8  
  
... 118: ac 03 01 02      STO  $3,base,offset
```



Anwendung

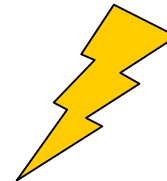
Zugriff auf Elemente eines Vektors

	LOC	Data_Segment
	GREG	@
A	OCTA	101, 102, 103, 104, 105, 106, 107, 108
N	OCTA	8
Sum	OCTA	
	LOC	#100
Main	...	



Adressierung

LOC	Data_Segment	
(\$254=#20000000 00000000)	GREG	@
2000000000000000:	A	OCTA 101,102,103,104,105,106,107,108
...000: 00000000		
...004: 00000065	= 101	: A ₀
...008: 00000000		
...00c: 00000066	= 102	: A ₁
...		
...038: 00000000		
...03c: 0000006c	= 108	: A ₇
...040: 00000000	N	OCTA 8
...044: 00000008		
...048: 00000000	Sum	OCTA
...04c: 00000000		



Index <> Offset!



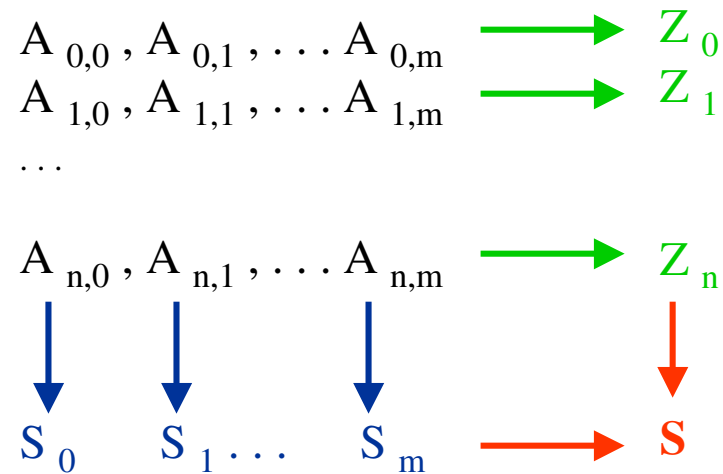
Beispiel : Summierung der Vektor - Elemente

Main	SET	s,0	Summe
	SET	i,0	Index
	LDO	n,N	
	LDA	base,A	Basis fest
	SET	offset,0	
Wdhl	LDO	ai,base,offset	indizierte Adressierung
	ADD	s,s,ai	Summe
	ADD	offset,offset,8	Offset variabel
	ADD	i,i,1	Index
	CMP	test,i,n	Ende ?
	BN	test,Wdhl	
	STO	s,Sum	
	TRAP	0,Halt,0	



Aufgabe

Gegeben eine Matrix **A**



Berechnung der Zeilensummen Z_0, Z_1, \dots, Z_n
Spaltensummen S_0, S_1, \dots, S_m
Gesamtsumme **S**



Aufgabe

Kreditverwaltung

Ein Geldverleiher verwaltet seine Kunden nach folgendem Schema
Pro Ausleihe ein Datensatz

Datum	Name	Betrag
-------	------	--------

OCTA OCTA OCTA

- Vereinbaren Sie die entsprechenden Daten für ein Beispiel im Datensegment
- Schreiben Sie ein MMIX – Programm zur Bestimmung
 - der gesamten Aussenstände
 - des größten Schuldenbetrags



Kreditverwaltung - Datenstrukturen

LOC Data_Segment
GREG @

1. Datensatz → Kredit
BYTE "18.01.06" Datum
BYTE "Huber" Name
OCTA 1267 Betrag

2. Datensatz →
BYTE "19.06.05"
BYTE "Maier"
OCTA 856



Längster Name
8 Byte !

3. Datensatz →
...
BYTE "01.11.05"
BYTE "Bauer"
OCTA 87

...
OCTA 0,0,0 Ende Daten

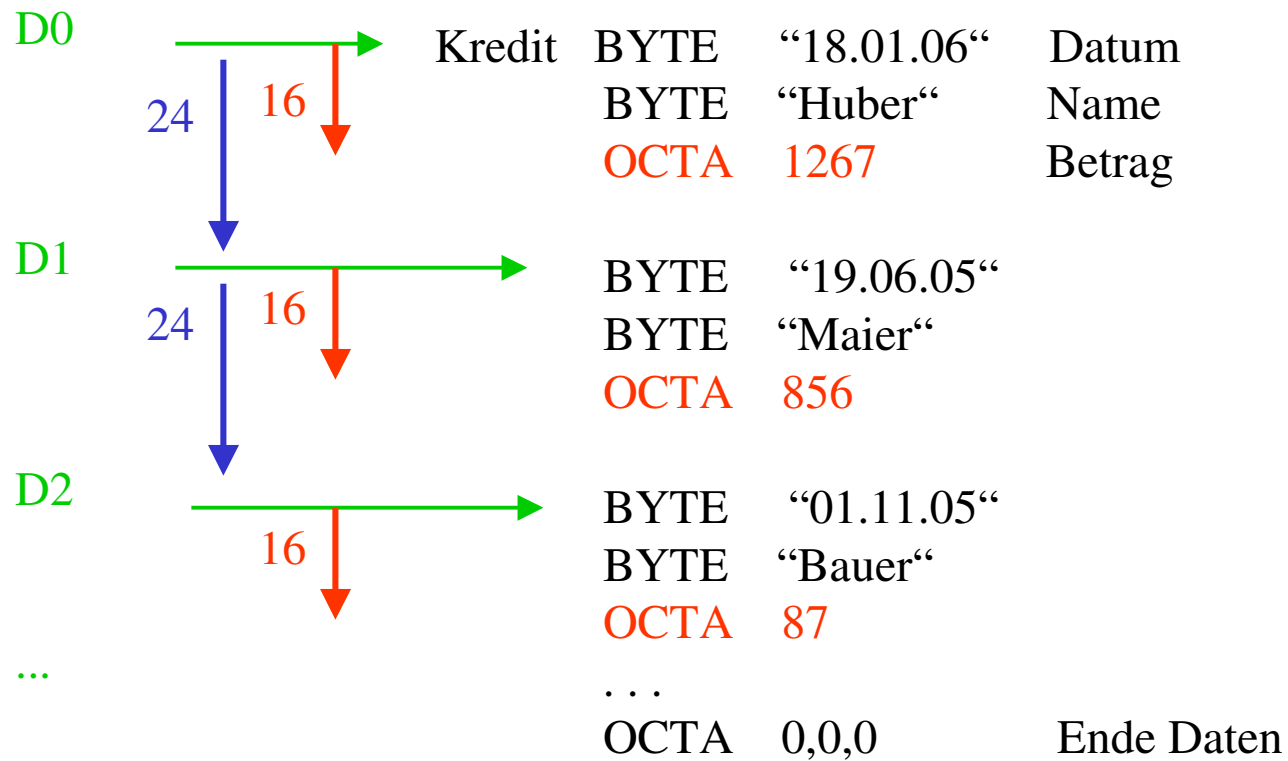


Kreditverwaltung - Datenstrukturen

2000 0000 0000 0000:	Kredit	BYTE	“18.01.06“	Datum
...000:	31 38 2e 30			
...004:	31 2e 30 31			
...008:	48 75 62 65	BYTE	“Huber“	Name
...00c:	72			
...010:	00 00 00 00	OCTA	1267	Betrag
...014:	00 00 03 58			
...018:	31 39 2e 30	BYTE	“19.06.05“	
...01c:	36 2e 30 35			
...020:	4d 61 69 65	BYTE	“Maier“	
...024:	72			
...028:	00 00 00 00	OCTA	856	
...02c:	00 00 03 58			



Adressierung





Gesamtaussenstände

Offset fest, Basis variabel

Main	LDA	base,Kredit	
	SET	offset,16	offset fixed
	SET	sum,0	
Loop	LDO	credit,base,offset	read credit
	BZ	credit,End	
	ADD	sum,sum,credit	
	ADD	base,base,3*8	next data block
	JMP	Loop	
End	STO	sum,Summe	
	TRAP	0,Halt,0	



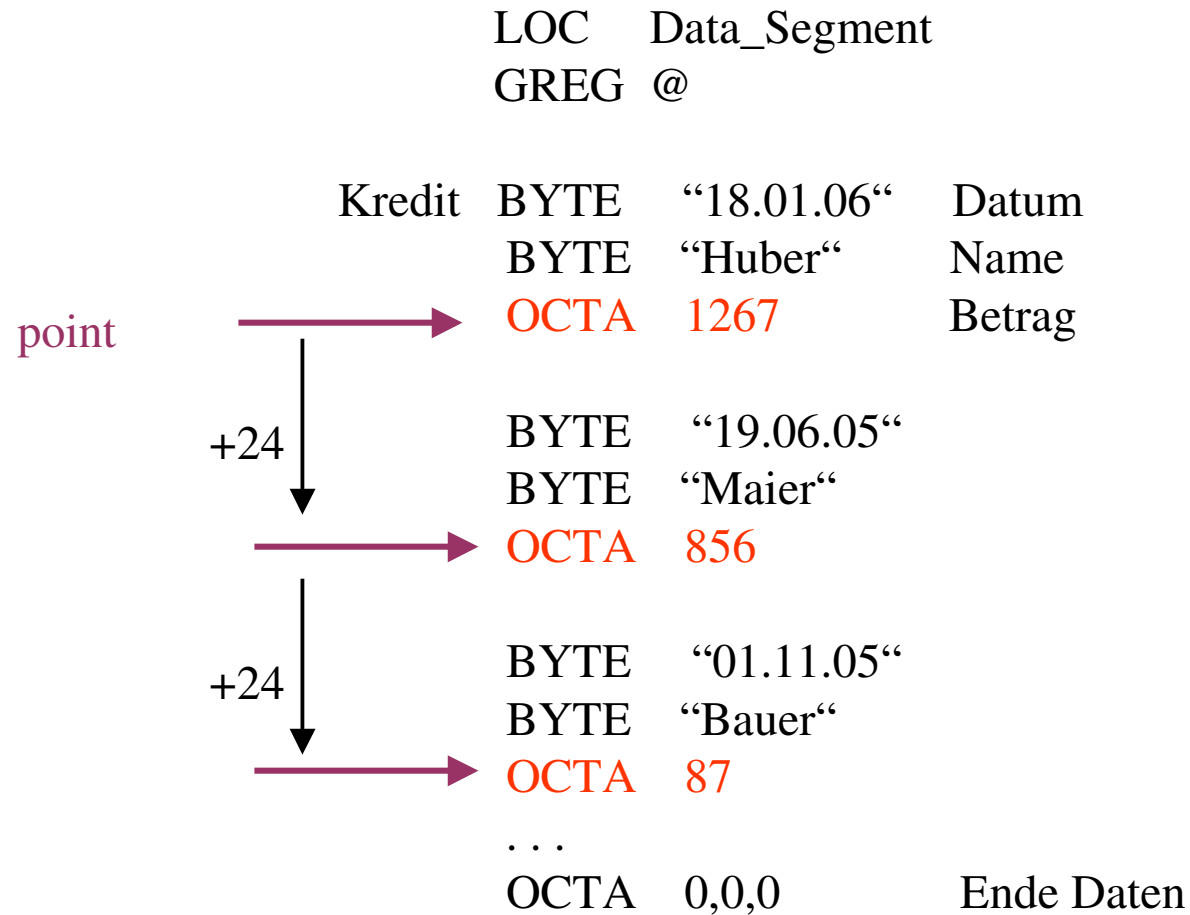
Gesamtaussenstände

Absoluter Pointer

Main	LDA	point,Kredit+16	
	SET	sum,0	
Loop	LDO	credit,point,0	read credit
	BZ	credit,End	
	ADD	sum,sum,credit	
	ADD	point,point,3*8	next data block
	JMP	Loop	
End	STO	sum,Summe	
	TRAP	0,Halt,0	



Adressierung



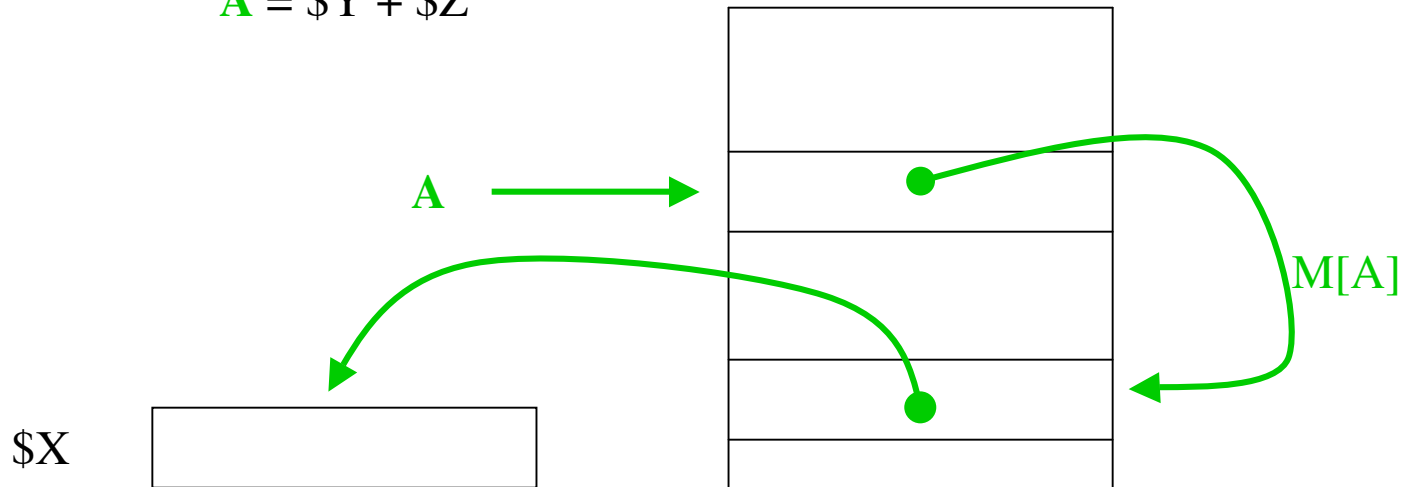


Speicher - indirekte Adressierung

LDO \$X,\$Y,\$Z

M[A] enthält die Adresse des Operanden

A = \$Y + \$Z



$\$X = M[M[A]]$

Bei MMIX nicht realisiert !



Speicher - indirekte Adressierung

Wichtige Anwendung : **verkettete Liste**

Explizite Programmierung von Hand



Siehe nächste Vorlesung

