

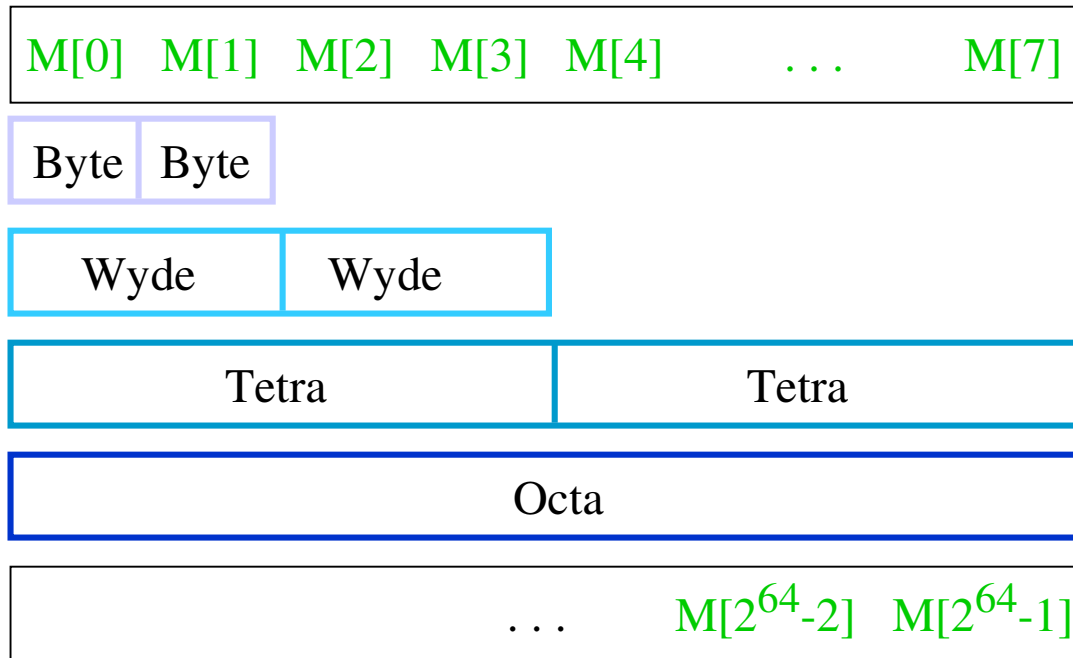


Alignment

- Ausrichtung von Daten im Datensegment
- Ausrichtung von Daten bei LOAD/STORE – Befehlen



Der Arbeitsspeicher von MMIX



Zugriff nur strukturiert möglich



Zugriff nur strukturiert möglich

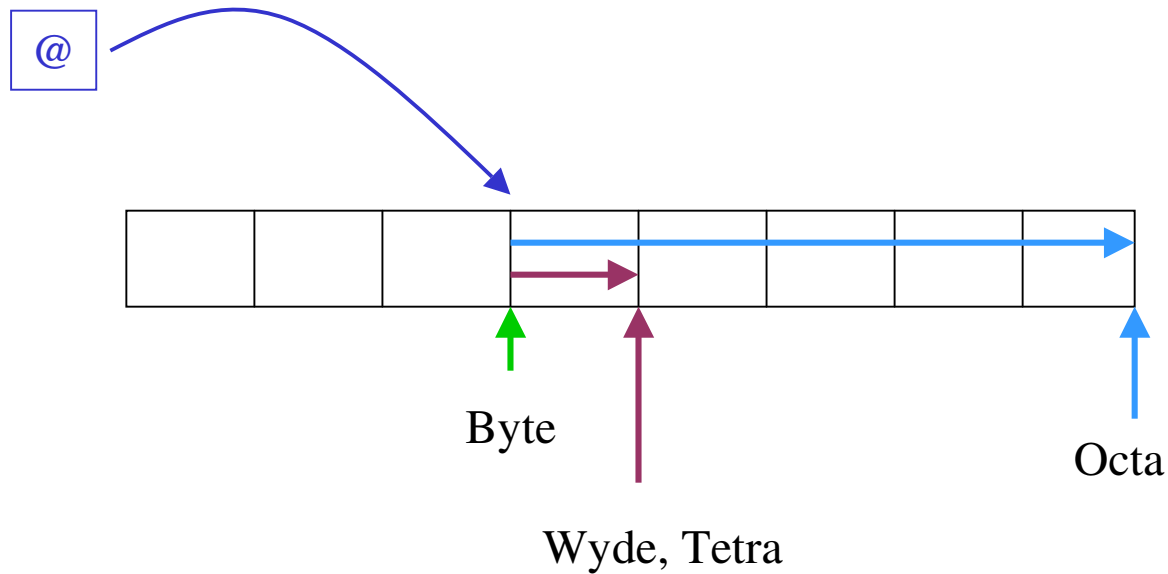
- Byte – Adresse : beliebig
2000 ... 000**0** #2000 ... 000**1** ... # 2000 ... 000**f**
2000 ... 001**0** ...
- Wyde – Adresse : gerade
2000 ... 000**0** #2000 ... 000**2** ... # 2000 ... 000**e**
2000 ... 001**0** ...
- Tetra – Adresse : teilbar durch 4
2000 ... 000**0** #2000 ... 000**4** ... # 2000 ... 000**c**
2000 ... 001**0** ...
- Octa – Adresse : teilbar durch 8
2000 ... 000**0** #2000 ... 000**8**
2000 ... 001**0** ...



Alignment

➔ Ausrichtung der Daten im Speicher durch den Assembler

Zuteilung der **nächsten passenden** Adresse
dazwischen “leere“ Speicherzellen (Aufrundung)



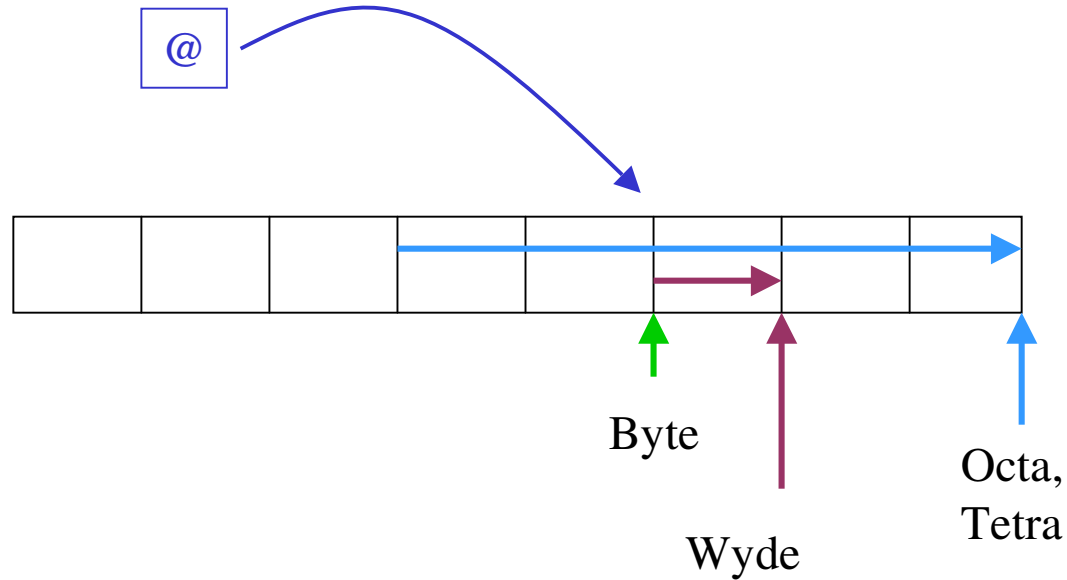


Alignment



Ausrichtung der Daten im Speicher durch den Assembler

Zuteilung der **nächsten passenden** Adresse
dazwischen "leere" Speicherzellen (Aufrundung)

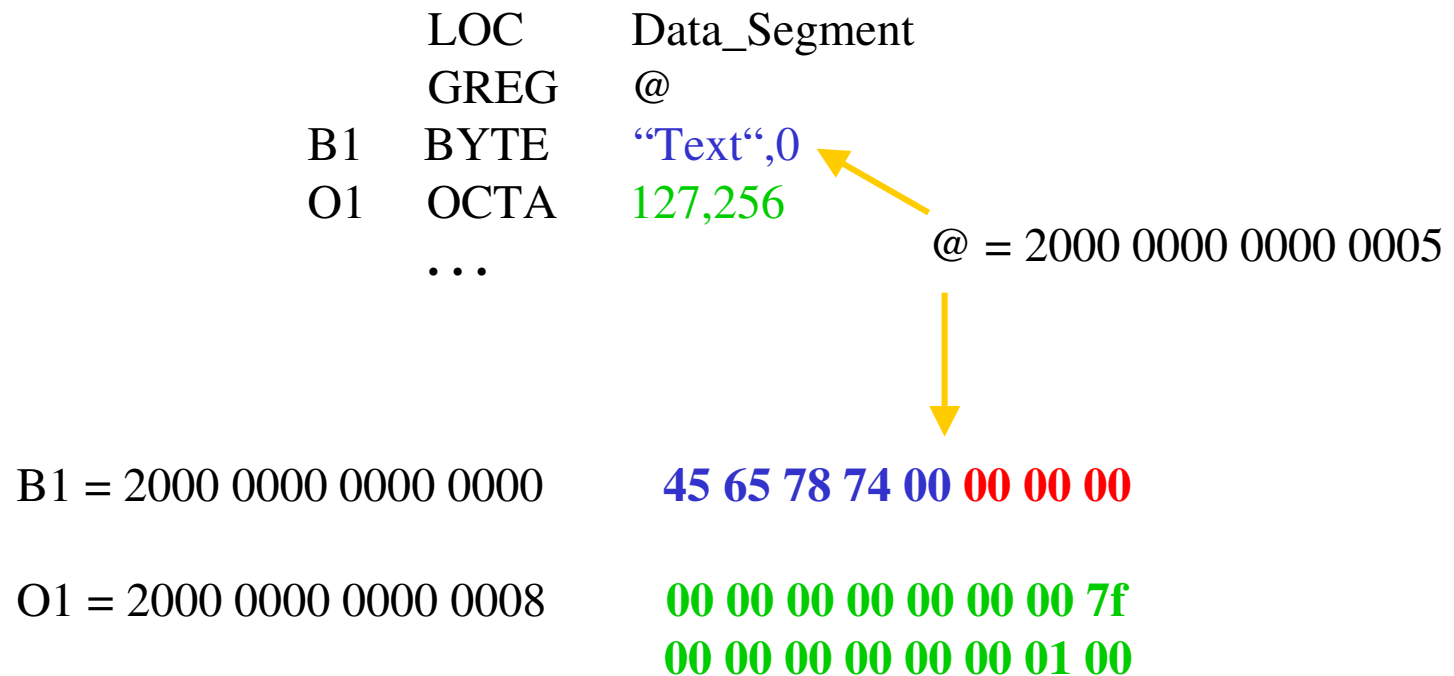




Alignment

Ausrichtung der Daten im Speicher durch den Assembler

Beispiel





Aufgabe

```
***      Ausrichtung      ***  
  
      LOC   Data_Segment  
      GREG  @  
  
Mybyte  BYTE  #11,#22,#33  
  
Mywyde  WYDE  #4444,#5555,#6666  
  
Mytetra TETRA #77777777,#88888888,#99999999  
  
Myocta  OCTA  #AAAAAAAAAAAAAAAAAAAA  
      ...
```



Alignment

Aufgabe

M 2000 0000 0000 0000 :



Alignment

Assembler – Listing Anzeige von TETRAs

```

                                LOC   Data_Segment
                                GREG  @
($254=#20000000 00000000)

... 000: 112233   Mybyte  BYTE  #11,#22,#33

... 004: 44445555 Mywyde  WYDE  #4444,#5555,#6666

... 00c: 77777777 Mytetra  TETRA #77777777,#88888888,#99999999
... 010: 88888888
... 014: 99999999

... 018: 0aaaaaaa Myocta   OCTA  #AAAAAAAAAAAAAAAAAAAA
... 01c: aaaaaaaa
```



Alignment

Assembler – Listing

Symbol table:

Main = #00000000000000100 (1)
Mybyte = #200000000000000000 (2)
Myocta = #2000000000000000018 (5)
Mytetra = #200000000000000000c (4)
Mywyde = #2000000000000000004 (3)



Alignment

Interpreter – Listing Anzeige von OCTAs

```
mmix>
0 instructions, 0 mems, 0 oops; 0 good guesses, 0 bad
(now at location #0000000000000100)

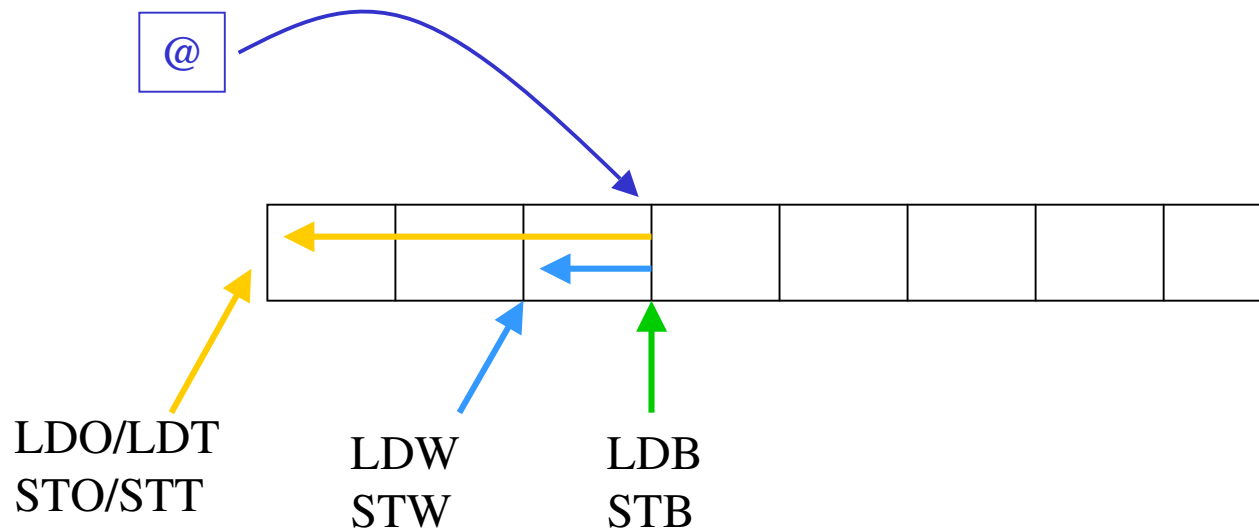
mmix> M2000000000000000# ... +4           “leere“ Zellen
                                           ↓
mmix> M8[#2000000000000000000]=#1122330044445555
                                           ↓
M8[#200000000000000000008]=#6666000077777777
                                           ↓
M8[#200000000000000000010]=#8888888899999999
                                           ↓
M8[#200000000000000000018]=#aaaaaaaaaaaaaaaa
                                           ↓
M8[#200000000000000000020]=#0
```



Ausrichtung von Daten bei LOAD/STORE – Befehlen

Abrunden der Adresse

LDO/STO		3
LDT/STT	Löschen der letzten	2 Bit
LDW/STW		1

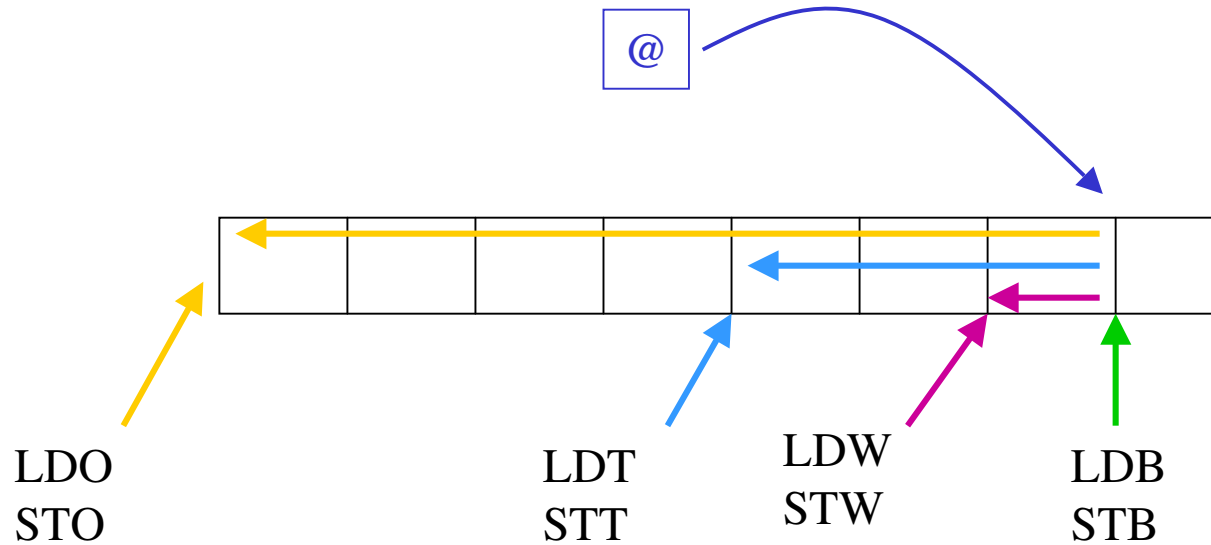




Ausrichtung von Daten bei LOAD/STORE – Befehlen

Abrunden der Adresse

LDO/STO		3
LDT/STT	Löschen der letzten	2 Bit
LDW/STW		1





Ausrichtung von Daten bei LOAD/STORE – Befehlen

LOAD – Befehle

LDB/LDBU ... LDO/LDOU

- Berechnung der Adresse
- Übernahme des Operanden rechtsbündig in Register
- dabei Auffüllen mit 0 (unsigned) bzw. Vorzeichen (signed)
- Keine Overflow - Anzeige



Ausrichtung von Daten bei LOAD/STORE – Befehlen

Beispiel

```
LOC  Data_Segment
GREG @
```

```
Myocta OCTA #1122334455667788,#99aabbccddeeff00
```

```
2000000000000000:
```

```
...000: 11 22 33 44
```

```
...004: 55 66 77 88
```

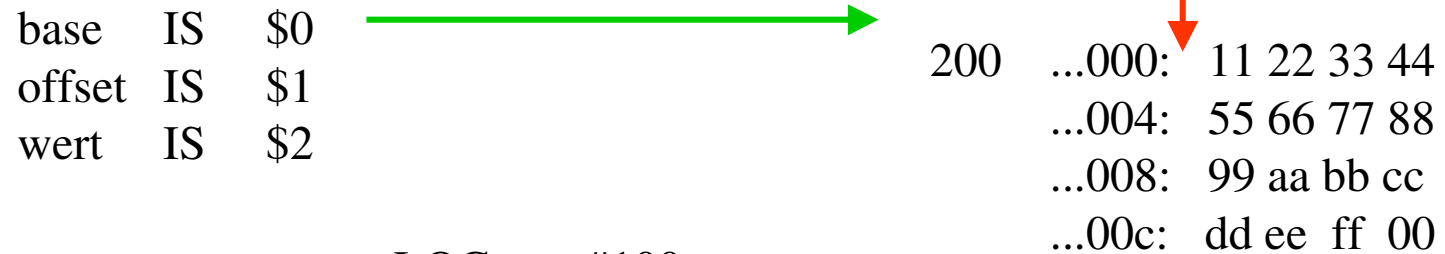
```
...008: 99 aa bb cc
```

```
...00c: dd ee ff 00
```



Alignment

Beispiel LOAD unsigned



```
LOC #100
Main LDA base,Myocta
      SET offset,0
      LDBU wert,base,offset
      LDWU wert,base,offset
      LDTU wert,base,offset
      LDOU wert,base,offset
```



Alignment

Beispiel LOAD unsigned

200 ...000: ↓ 11 22 33 44
...004: 55 66 77 88
...008: 99 aa bb cc
...00c: dd ee ff 00

(LDBU) \$2 = 1[2] = M[#2000000000000000+#0] = #0000 0000 0000 00**11**

(LDWU) \$2 = 1[2] = M[#2000000000000000+#0] = #0000 0000 0000 **1122**

(LDTU) \$2 = 1[2] = M[#2000000000000000+#0] = #0000 0000 **1122 3344**

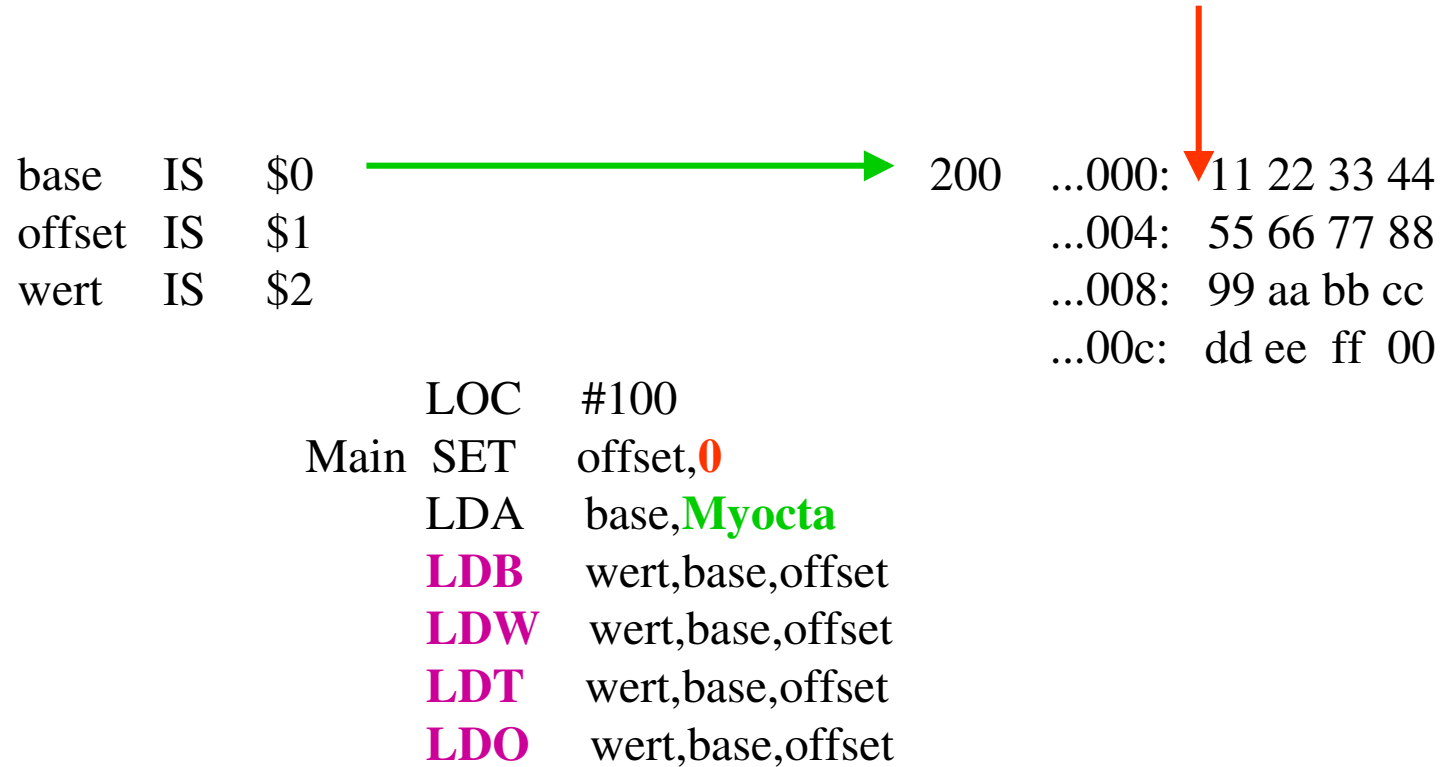
(LDOU) \$2 = 1[2] = M[#2000000000000000+#0] = **#1122 3344 5566 7788**

Hexadezimale ↑
Anzeige !



Alignment

Beispiel LOAD signed, positiv





Alignment

Beispiel LOAD signed, positiv

200	...000:	11 22 33 44
	...004:	55 66 77 88
	...008:	99 aa bb cc
	...00c:	dd ee ff 00

(LDB) \$2 = l[2] = M[#20000000000000000+#0] = 17
 mmix> l[2] = #0000 0000 0000 00**11**

(LDW) \$2 = l[2] = M[#20000000000000000+#0] = 4386
 mmix> l[2] = #0000 0000 0000 **1122**

(LDT) \$2 = l[2] = M[#20000000000000000+#0] = 287454020
 mmix> l[2] = #0000 0000 **1122 3344**



(LDO) \$2 = l[2] = M[#20000000000000000+#0] = 1234605616436508552
 mmix> l[2] = **#1122 3344 5566 7788**

Arithmetische
Anzeige !



Alignment

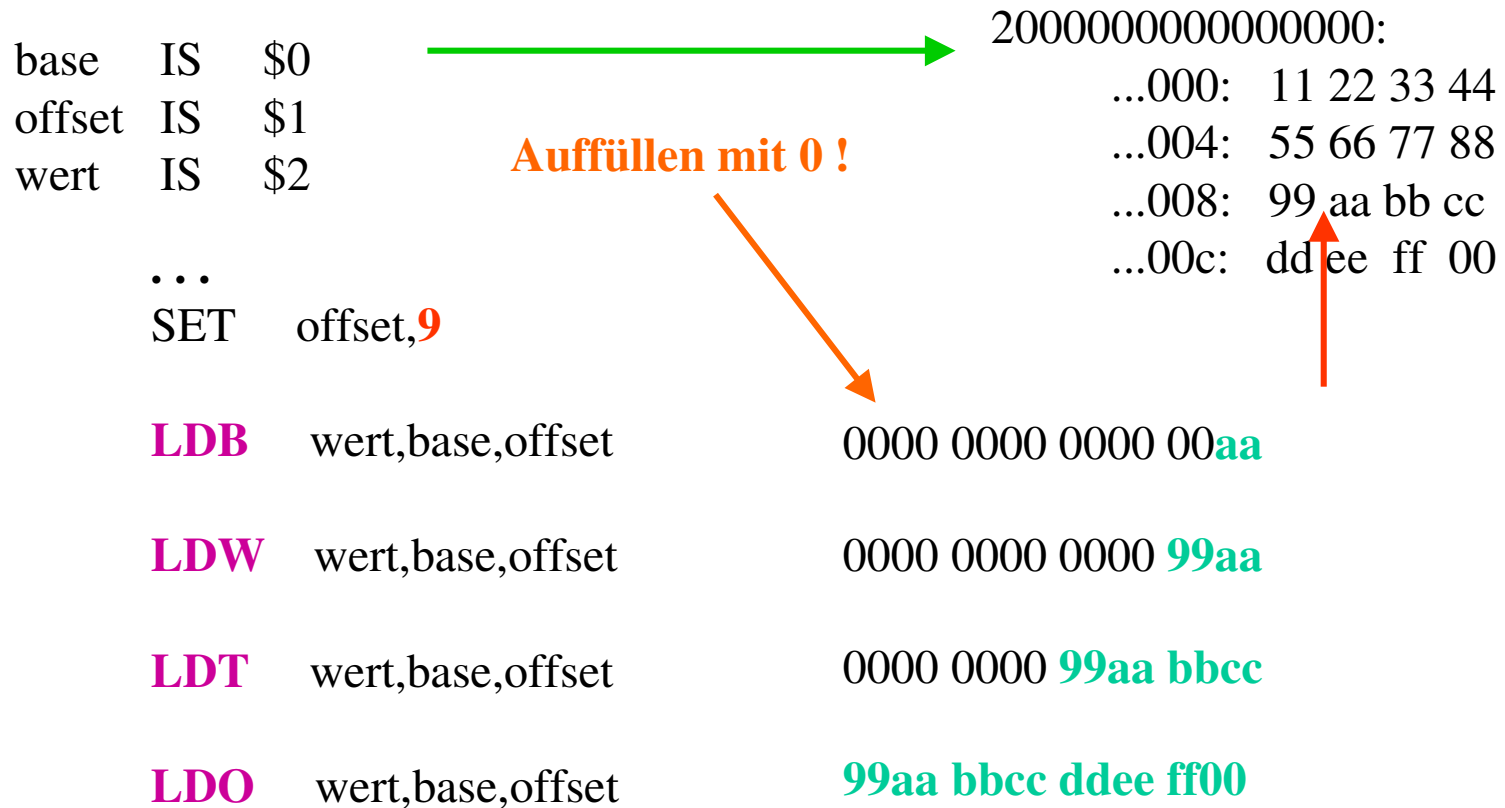
Beispiel LOAD unsigned/signed positiv

base	IS	\$0		200000000000000000:	
offset	IS	\$1		...000:	11 22 33 44
wert	IS	\$2		...004:	55 66 77 88
...				...008:	99 aa bb cc
...				...00c:	dd ee ff 00
SET		offset,3			
LDBU		wert,base,offset		0000 0000 0000 00	44
LDWU		wert,base,offset		0000 0000 0000 33	44
LDTU		wert,base,offset		0000 0000 1122	3344
LDOU		wert,base,offset		1122 3344 5566 7788	



Alignment

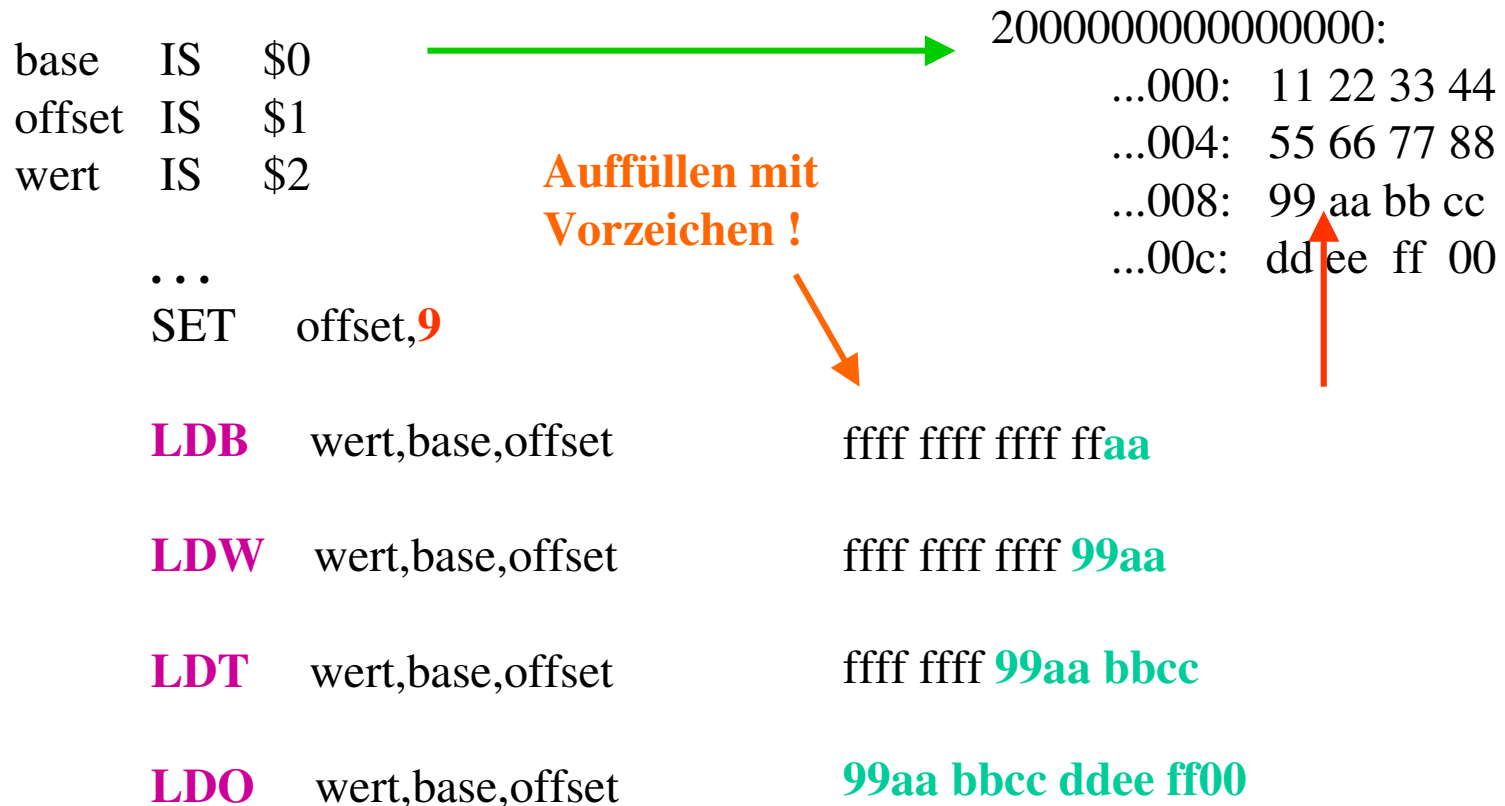
Beispiel LOAD unsigned





Alignment

Beispiel LOAD signed





Alignment

Beispiel LOAD signed, negativ

(LDB) \$2 = l[2] = M[#2000000000000000+#9] = -86
mmix> l[2] = #ffff ffff ffff ffaa

(LDW) \$2 = l[2] = M[#2000000000000000+#9] = -26198
mmix> l[2] = #ffff ffff ffff 99aa

(LDT) \$2 = l[2] = M[#2000000000000000+#9] = -1716864052
mmix> l[2] = #ffff ffff 99aa bbcc

(LDO) \$2 = l[2] = M[#2000000000000000+#9] = -7373874951294615808
mmix> l[2] = #99aa bbcc ddee ff00

Arithmetische
Anzeige !

rA = 0 : kein Overflow



Ausrichtung von Daten bei LOAD/STORE – Befehlen

STORE – Befehle

STB/STBU ... STO/STOU

- Berechnung der Adresse im Speicher wie bei Load
- Übernahme des Register – Operanden in den Speicher
- dabei Overflow – Anzeige möglich (signed)



Alignment

Beispiel STORE – Befehl

LOC Data_Segment
GREG @

Myocta OCTA #1122334455667788
Test OCTA 0,0

base IS \$0
offset IS \$1
wert IS \$2
Zero IS \$3

= 1234605616436508552

Testmuster



Alignment

Beispiel STORE – Befehl

Main	LDO	wert, Myocta	200 ...000:	11 22 33 44
	LDA	base, Test	...004:	55 66 77 88
			...008:	00 00 00 00
			...00c:	00 00 00 00
	SET	offset, 0		
	STBU	wert, base, offset		
	STO	zero, Test		
	SET	offset, 7		
	STBU	wert, base, offset		
	STO	zero, Test		



Alignment

STORE – Befehl, **unsigned**

Testmuster in \$2

\$2 = #11223344556677 **88**

Selektiertes
Byte in \$2

SET offset, **0**

STBU wert, base, **offset**

(STBU) M1[#20000000000000008+#**0**] = 1234605616436508552

M8[#20000000000000008] = #**88**00 0000 0000 0000

Selektiertes Byte im Speicher

keineOverflow - Anzeige!



Alignment

STORE – Befehl, unsigned

Testmuster in \$2



\$2 = #11223344556677 **88**

Selektiertes Byte in \$2



SET offset, **7**

STBU wert, base, **offset**

(STBU) M1[#20000000000000008+#**7**] = 1234605616436508552

M8[#20000000000000008] = **#88**

Selektiertes Byte im Speicher



= #0000 0000 0000 00**88**

Keine Overflow - Anzeige!



Alignment

STORE – Befehl, signed

Testmuster in \$2

\$2 = #11223344556677 **88**

Selektiertes Byte in \$2

SET offset,0

STB wert,base,offset

(STB) M1[#2000000000000008+#0] = 1234605616436508552 ,

M8[#2000000000000008] = #8800 0000 0000 0000

= -8646911284551352320

Selektiertes Byte im Speicher

rA = # 00040
Overflow !



Alignment

STORE – Befehl, signed

SET offset,7

STB wert,base,offset

(STB) M1[#20000000000000008+#0] = 1234605616436508552

M8[#20000000000000008] = #88

Selektiertes Byte im Speicher

= #0000 0000 0000 0088

Testmuster in \$2

#11223344556677

88

Selektiertes Byte in \$2

rA = # 00040
Overflow !