Maschinenorientierte Programmierung

PC Inside 2021 Aufgaben Name:_____

Adressierung von Daten:
:

1. Der Inhalt der Speicherplätze 8 bis 18H im Datensegment ist in die Speicherplätze 48H bis 58H zu kopieren. Verwenden Sie BX als Basiszeiger, SI als Indexzeiger und CX als Zählregister. Außerdem kann entweder mit Hilfe von DI oder SI und einer Verschiebungs-konstanten gearbeitet werden.

2. Die Speicherplätze 30H bis 40H im Datensegment sind zu löschen (auf Null zu setzen).

3. Suchen Sie mit einem Maschinenprogramm ab Speicherplatz 0 im Datensegment nach einem Speicherinhalt, der einem 'D' entspricht. Hinterlegen Sie die gefundene Speicherplatzadresse als Wort in 50H und 51H.

4. Es soll die Summe der Inhalte der Speicherplätze 28H bis 2FH gebildet und das Ergebnis in 30H abgelegt werden.

5. Zählen Sie die Anzahl der negativen Inhalte a) als Bytes b) als Worte in dem Datenblock von 28H bis 37H. Das Ergebnis soll an der Adresse 0 gespeichert werden.

6. Suchen Sie das größte a) Byte b) Wort in dem Datenblock von 8 bis 27H. Das Mamimum selbst soll im Speicherplatz 0, seine Adresse in 3 und 4 nach Ablauf des Programms zu finden sein.

7. Ersetzen Sie alle '*'-Zeichen im Datenblock 0 bis 27H durch Leerzeichen.

(Lösung kann mit PC Inside geprüft werden)

Befehlscodierung bei direkter und indirekter Adressierung:

Führen Sie eine vollständige Codierung der Befehle durch!
(Informationen dazu siehe nächste Seite)

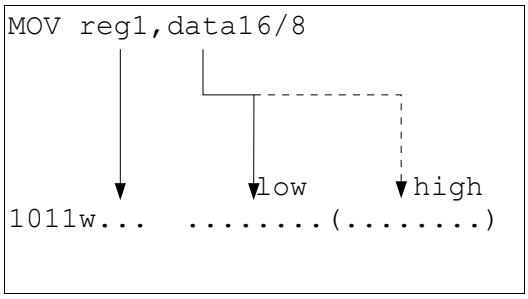
```
1. a) MOV CX,[BX][7]
   b) MOV AX,[BX][800H]
   c) MOV [SI],AX
   d) MOV [C800H],DI
   e) MOV SI,[DI]
   f) MOV [DI],SI
   g) MOV [BX][SI][A00AH],BP
   h) MOV SP,[DI][255]
```

```
2. a) MOV CS:[FOOOH],BX
   b) MOV CS:[BX][DI],SP
   c) MOV DS:[BP][40H],AX
   d) MOV CX,BP
   e) MOV DX,ES:[BP][SI]
   f) MOV SP,SS:[BP][FDH]
```

(Lösung kann mit PC Inside ermittelt werden)

Befehlsobjektcodes des MOV-Befehls bei verschiedenen Adressierungen:

Beispiel 1:



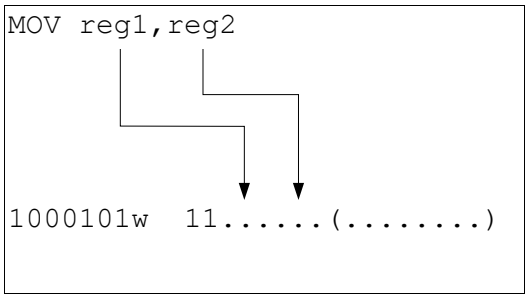
move data immediate into register

kopiere umittelbare Daten in ein Register

Allgemein:

```
Opcode|W|REG  ........(........)
```

Beispiel 1:



move data from 2nd to 1st register

kopiere Daten vom 2. ins 1. Register

Allgemein:

```
Opcode|D|W| MOD|REG|R/M (........(........))
```

Abkürzungen:

D	destination	Ziel
W	word or byte	Wort oder Byte
MOD	mode	Modus
REG	register	Register
R/M	register or memory	Register oder Speicher

MOD-Feld und R/M-Feld:

MOD:	**00**	**01**	**10**	**11**
R/M				**w: 1 oder 0**
000	BX + SI	BX + SI + data8	BX + SI + data16	000 für AX oder AL
001	BX + DI	BX + DI + data8	BX + DI + data16	001 für CX oder CL
010	BP + SI	BP + SI + data8	BP + SI + data16	010 für DX oder DL
011	BP + DI	BP + DI + data8	BP + DI + data16	011 für BX oder BL
100	SI	SI + data8	SI + data16	100 für SP oder AH
101	DI	DI + data8	DI + data16	101 für BP oder CH
110	data16	BP + data8	BP + data16	110 für SI oder DH
111	BX	BX + data8	BX + data16	111 für DI oder BH

DARSTELLUNG VON INFORMATION

Alle Computer verarbeiten Informationen, welche aus Zahlen und Texten bestehen. Hier wird nun die interne Darstellung von Informationen auf einem Computer besprochen.

Interne Darstellung von Information

Information wird in einem Computer in Gruppen von Bits gespeichert. Das Wort *Bit* steht für engl. *Binary digit*, was soviel wie zweiwertige Ziffer bedeutet. Die elektronische Darstellung erfolgt durch die Zustände „ungeladen“ oder „aus“ und „geladen“ oder „ein“ und entspricht einer Logik mit zwei Zuständen. Diese Zustände werden logisch durch die Symbole „0" und „1" dargestellt. Solche elektronischen Schaltungen werden *binäre logische Schaltkreise* genannt. Auf einem Computer wird die gesamte Informationsverarbeitung im binären Format durchgeführt wird. Eine Gruppe von acht Bits wird *Byte* genannt.

Zwei Dinge müssen innerhalb des Computers in diesem binären Format dargestellt werden: Das Programm, das eine Folge von Befehlen ist, und die Daten, mit denen das Programm arbeitet. Diese binären Daten können somit Befehle sein, Zahlen darstellen oder auch kodierter Text sein. Hier soll die Darstellung mit acht Bits besprochen werden.

Darstellung numerischer Daten

Ganze Zahlen können *direkt binär* dargestellt werden. Diese direkte binäre Codierung ist einfach die Darstellung des dezimalen Wertes einer Zahl im binären System. Im binären System repräsentiert das am weitesten rechts stehende Bit (Bit 0) den Wert $2^0 = 1$. Das nächste Bit links davon (Bit 1) repräsentiert den Wert $2^1 = 2$; das folgende $2^2 = 4$ und das am weitesten links stehende Bit (Bit 7) den Wert $2^7 = 128$. Zum Beispiel stellt
10011110 den Wert

$$1 \cdot 2^7 + 0 \cdot 2^6 + 0 \cdot 2^5 + 1 \cdot 2^4 + 1 \cdot 2^3 + 1 \cdot 2^2 + 1 \cdot 2^1 + 0 \cdot 2^0 =$$
$$128 + 16 + 8 + 4 + 2 = 158$$

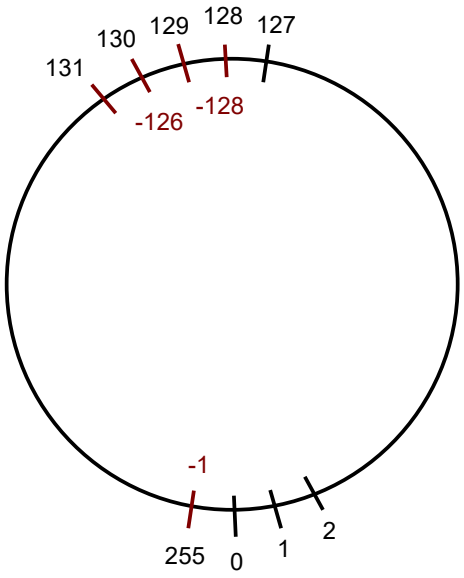
dar.

Mit acht Bits ist es deshalb möglich, direkt die postiven Zahlen 00000000 bis 11111111 (0 bis 255) darzustellen. Es liegen hier ausschließlich positive Zahlen von 0 bis 255 vor. Man möchte aber auch mit negativen Zahlen rechnen.
.

Binärzahlen mit Vorzeichen

Bei der binären Darstellung mit Vorzeichen gibt das Bit ganz links das Vorzeichen der Zahl an. Üblicherweise wird 0 benutzt, um eine *positive* Zahl zu kennzeichnen und 1 wird benutzt, um eine *negative* Zahl zu kennzeichnen. Die Anordnung kann grafisch durch den Zahlenkreis der 8-Bit-Zahlen anschaulich gemacht werden.

Zahlenkreis der 8-Bit-Zahlen



Zum Beispiel stellt 11111111 den Wert -1 dar, während 01111111 den Wert +127 darstellt. Wir können nun positive und negative Zahlen darstellen, aber wir haben die maximale Größe dieser Zahlen auf 127 (bzw. 128) verringert.

Als ein weiteres Beispiel entspricht 00000001 dem Wert 1 (die führende 0 ist „+ "), gefolgt von 0000010 = 2. Geht man nun von der 0 einen Schritt rückwärts, kommt nan zur -1 und im Zahlenkreis der 8-Bit-Binärzahlen zu 11111111. Im Folgenden wird an jede binär dargestellte Zahl ein „B“ angehängt.

Der Übergang von einer positiven Zahl in Binärsarstellung zur ihrer Gegenzahl in binärdarstellung erhält man mittels **Zweier-Komplement**.

Das Einer-Komplement
Das einer-Komplement der Zahl 3 = 00000011B erhält man, indem man jedes Bit komplementiert, also alle Nullen durch Einsen ersetzt und alle Einsen durch Nullen: 11111100B. Dies entspricht allerdings beim Zahlenkreis der Zahl -4 und nicht der Zahl -3.

Zweier-Komplement-Darstellung

In der Zweier-Komplement-Darstellung werden positive Zahlen wie üblich binär mit Vorzeichen dargestellt, ebenso wie beim Einer- Komplement. Der Unterschied liegt in der Darstellung *negativer Zahlen.* Die Darstellung einer negativen Zahl im Zweier-Komplement erhält man, indem man zunächst das Einer-Komplement berechnet und dann *eins addiert.*

Beispiel:
+3 wird in binärer Darstellung mit Vorzeichen durch 00000011 dargestellt. Das Einer-Komplement davon ist 11111100. Das Zweier- Komplement erhält man durch Addition eine· r Eins. Es ist 11111101. Dies entspricht beim Zahlenkreis -3.

+5 wird in binärer Darstellung mit Vorzeichen durch 00000101 dargestellt. Das Einer-Komplement davon ist 11111010. Das Zweier-Komplement erhält man durch Addition einer Eins. Es ist 11111011. Dies entspricht beim Zahlenkreis -5.

Wir berechnen nun 3 – 5 = 3 + (-5):

```
  00000011
+ 11111011
----------
  11111110
```

Das Ergebnis lautet richtig 11111110B = -2.

Das Übertragsbit C (Carry-Flag)

Findet bei der Addition ein Übertrag von Bit 7 auf das nicht existierende Bit 8 statt, spricht man von einem Übertrag. Findet bei der Subtraktion ein Borgen vom nicht existierenden Bit 8 statt, spricht man ebenfalls von einem Übertrag. Dieser wird mangels ausreichender Datenbreite als Übertragsbit angezeigt. Bei einem Mikroprozessor wird das sogenannten Carry-Flag auf 1 gesetzt.

Das Überlaufbit O (Overflow-Flag)

Eine Überschreitung der Stelle zwischen 127 und -128 auf dem Zahlenkreis bei Addition oder Subtraktion stellt einen Überlauf dar - während eine Überschreitung der Stelle zwischen 255 und 0 auf dem Zahlenkreis bei Addition oder Subtraktion den vorher genannten Übertrag darstellt.

Ein Überlauf tritt in folgenden vier Situationen auf:

1. bei der Addition großer positiver Zahlen (70 + 80 = 150 = -106 auf dem Zahlenkreis),
1. bei der Addition stark negativer Zahlen (-70 + (-80) = - 150 = + 106 auf dem Zahlenkreis ,
2. bei der Subtraktion einer großen positiven von einer stark negativen Zahl (-120 – 80 = -200 = +56 auf dem Zahlenkreis),
3. bei der Subtraktion einer stark negativen Zahl von einer großen positiven Zahl (120 – (-80) = +200 = -56 auf dem Zahlenkreis),.

Technisch wird die Überlaufanzeige, das Overflow-Flag, gesetzt, wenn es einen Übertrag von Bit. 6 nach Bit 7 und keinen externen Übertrag (von Bit 7 nach Bit 8) gibt. Es wird ebenfalls gesetzt, wenn es keinen Übertrag von Bit 6 nach Bit 7, aber einen externen Übertrag (von Bit 7 nach Bit 8) gibt. Das gibt an, dass sich Bit 7, das Vorzeichen des Ergebnisses, versehentlich geändert hat. Also: Gesetzt, wenn entweder Übertrag von Bit 6 nach Bit 7 oder von Bit 7 nach Bit 8 erfolgt..

Ein Übertrag sowie ein Überlauf zeigt an, dass das Ergebnis einer Addition oder Subtraktion mehr Bits erfordert als in dem Standard-8-Bit-Register, das das Ergebnis enthalten soll, zur Verfügung stehen.

Der Übertrag und der Überlauf

Das Übertrags- und das Überlaufsbit heißen *Kennzeichen (Flags).* Sie sind in jedem Mikroprozessor vorgesehen und dienen dem effizienten Programmieren. Diese beiden Anzeigen befinden sich in einem besonderen Register, dem Kennzeichen- oder „Status"-Register. Dieses Register enthält noch weitere Anzeigen.

Beeinflusste Kennzeichen der einzelnen Befehle (nicht vollständig)

Befehl	***O***	***D***	***I***	***T***	***S***	***Z***	***A***	***P***	***C***
ADD	+				+	+	+	+	+
AND	0				+	+	?	+	0
CLC	-				-	-	-	-	0
CMC	-				-	-	-	-	+
CMP	+				+	+	+	+	+
DEC	+				+	+	+	+	-
DIV	?				?	?	?	?	?
INC	+				+	+	+	+	-
MOV	-				-	-	-	-	-
MUL	+				?	?	?	?	+
NEG	+				+	+	+	+	+
NOT	-				-	-	-	-	-
OR	0				+	+	?	+	0
STC	-				-	-	-	-	1
SUB	+				+	+	+	+	+
XOR	0				+	+	?	+	0

Z	Zero-Flag
C	Carry-Flag
S	Signum-Flag
O	Overflow-Flagt
P	Parity-Flag

+	Flag verändert
-	Flag unverändert
1	Flag gesetzt
0	Flag zurückgesetzt
?	Flag undefiniert