

Weihnachts-Übungsblatt - Lösung

1. Umrechnen

a) zwischen verschiedenen Zahlensystemen

Basis 2	3	7	8	13	16 (0-F)
10101010	20022	332	252	101	AA
1001101	2212	140	115	5C	4D
100100011101	10012102	6542	4435	10A6	91D
111000110110	11222202	13415	7066	186B	E36
101000001	102220	636	501	1B9	141
10100111	20012	326	247	CB	A7

b) Komplementbildung zur obigen Tabelle:

Basis 2	3	7	8 (8 Z.)	13	16 (4 Z.)
01010110			126		56
0110011			51		33
011011100011			3343		6-E3
000111001010			77770412		FFF10A
010111111			277		BF
1011001			177531		FF59

c) Alle Rechenschritte sollen in einem 16 Bit Binärsystem erfolgen. (Ohne Umrechnung in Dez.)

(167+3638) Zwischenergebnis (Binär): 111011011101
 *4 ZE (Bin): 11101101110100
 +13 ZE (Bin): 11101110000001
 shl, 2 (Siehe Befehlsblätter) ZE: 1110111000000100
 -AE ZE (Bin): 1110110101010110
 shr, 1 ZE (Bin): 0111011010101011
 /30 ZE (Bin): 1111110100 Rest: 10011

d) Alle Rechenschritte sollen im Hexadezimalsystem erfolgen. (Ohne Umrechnung in Dezimal!)

(FF+FF) ZE (Hex): 1FE
 +FEFA ZE (Hex): 100F8
 *A3 ZE (Hex): A39DE8
 /4 ZE (Hex): 28E77A
 -FFFFFFE ZE (Hex): -D71884

e) Wie viele Operationen verarbeitet eine CPU mit 2,5GHz in 7 Minuten?

7 Minuten = 420 Sekunden, 2,5 Ghz = $2,5 \cdot 10^9$ Hz, $420s \cdot 2,5 \cdot 10^9$ Hz = 1050000000000

f) Wie hoch ist der Beschleunigungsfaktor wenn man die Aufgabe parallelisiert und 30% von einer zweiten, gleich schnellen CPU verarbeitet werden?

1050000000000 operationen aufteilen: 70%=735000000000, 30%=315000000000,

$315000000000 / 2,5 \cdot 10^9$ Hz = 126s, $735000000000 / 2,5 \cdot 10^9$ Hz = 294s

beide laufen parallel, also bei 294s fertig => Faktor=294s/420s=1,4285 => 43 % schneller!

2. Fachbegriffe

Erläutern Sie folgende Begriffe

EVA Prinzip	Register
Stack	Bus
Indirekte Adressierung	Harvard-Architektur
Befehlszyklus	Rechenwerk
Leitwerk	Steuerleitung
Sprungbefehl	E/A Werk
Interrupt	Befehlszähler
Ablaufsteuerung	Flags
Unterprogramm	Assembler

=> SIEHE SKRIPT, bzw. WIKIPEDIA.DE

3. Programmiertechnik

a) Schreiben Sie ein Assemblerprogramm, das die Zahlen von 10 bis 0 im Speicher mittels des Bubble Sort (Siehe Wikipedia.de) Algorithmus aufsteigend sortiert.

b) Erstellen Sie Flussdiagramme zu folgenden Assemblerprogrammen:

- Vergleich zweier einzugebender Wörter
- Quick Sort (Siehe Wikipedia.de)

c) Erläutern Sie den Unterschied zwischen:

- Interrupt, Makro und Unterprogramm

Interrupt: unterbricht aktuelles Programm komplett, Befehlszähler und Flags auf Stack gesichert, Rücksprung mit IRET der alles wieder vom Stack holt, langsamer als ein Unterprogrammaufruf

Makro: kein Stack, da der Code nur kopiert wird => größerer Code, schnell weil keine Rücksprünge nötig, Parameterübergabe sehr einfach

Unterprogramm: EINE Kopie im Speicher (auch bei mehreren Aufrufen) => Speicherersparnis, langsamer als Macros wegen Stackoperationen (Rücksprünge usw), können einfach aus anderen Programmiersprachen aufgerufen werden

- Zugriffen auf den Hauptspeicher und externe I/O Geräte
- Den Befehlen IRET und RET.

IRET ist der Rücksprung aus einem Interrupt, RET ist der Rücksprung aus einem Unterprogramm

4. Assembler

The screenshot shows the DOSBox 0.74-2 interface with the following components:

- Menu Bar:** File, View, Run, Breakpoints, Data, Options, Window, Help.
- Assembly Code:**

```

cs:0013>8AD0      mov     dl,al
cs:0015  B610      mov     dh,10
cs:0017  F6F6      div     dh
cs:0019  8ACC      mov     cl,ah
cs:001B  32E4      xor     ah,ah
cs:001D  B302      mov     bl,02
cs:001F  F6E3      mul     bl
cs:0021  BE0000     mov     si,0000
cs:0024  03C6      add     ax,si
cs:0026  8BF0      mov     si,ax
cs:0028  8B04      mov     ax,[si]
cs:002A  FEC1      inc     cl
cs:002C  D3D0      rcl     ax,cl
cs:002E  73D5      jnb     0005
cs:0030  50        push    ax
cs:0031  B402      mov     ah,02
cs:0033  CD21      int     21
cs:0035  58        pop     ax
cs:0036  EBCD      jmp     0005
cs:0038  B44C      mov     ah,4C
cs:003A  B000      mov     al,00
cs:003C  CD21      int     21
cs:003E  0000      add     [bx+sil],al
cs:0040  0000      add     [bx+sil],al
cs:0042  0000      add     [bx+sil],al
cs:0044  0000      add     [bx+sil],al
cs:0046  0000      add     [bx+sil],al
cs:0048  0000      add     [bx+sil],al
cs:004A  0000      add     [bx+sil],al
cs:004C  0000      add     [bx+sil],al
cs:004E  0000      add     [bx+sil],al
cs:0050  0000      add     [bx+sil],al
cs:0052  0000      add     [bx+sil],al
cs:0054  0000      add     [bx+sil],al

```
- Register Window (Right):**

```

ax 0074  c=0
bx 0000  z=1
cx 0000  s=0
dx 0000  o=0
si 0000  p=1
di 0000  a=0
bp 0000  i=1
sp 0000  d=0
ds 56BA
es 56AA
ss 56BA
cs 56BC
ip 0013

```
- Memory Window (Bottom):**

```

ds:0000 00 00 00 00 02 80 C0 FF 00 00
ds:0008 FF 7F E0 FF FF 7F E0 FF 00 00
ds:0010 00 00 00 00 00 00 00 00 00 00
ds:0018 00 00 00 00 00 00 00 00 00 00
ds:0020 B8 BA 56 8E D8 33 C0 33 00 00
ds:0028 DB 33 C9 33 D2 B4 08 CD 00 00
ds:0030 21 32 E4 8A D0 B6 10 F6 00 00
ds:0038 F6 8A CC 32 E4 B3 02 F6 00 00
ds:0040 E3 BE 00 00 03 C6 8B F0 00 00
ds:0048 8B 04 FE C1 D3 D0 73 D5 00 00
ds:0050 50 B4 02 CD 21 58 EB CD 00 00

```
- Footer:** F1-Help F2-Bkpt F3-Mod F4-Here F5-Zoom F6-Next F7-Trace F8-Step F9-Run F10-Menu

Was steht in den Registern und den Flags bei Erreichen des Breakpoints?

Reg	Wert	Flag	Wert
AX		C	
BX		Z	
CX		S	
DX		O	
SI		P	
SP			
IP			