Microcontroller

Selbststudium Semesterwoche 4

Aufgabe 2.5
Versuchen Sie den nachfolgenden Code zweilenweise in mnemonischen Code zumzusetzen. Alle Byteangaben sind hexadezimal.

```
$0829: A6 FF B7 0B B7 47 B7
$0830: 0F A6 01 B7 08 AE FF 6E FF 44 4E 0C 04 4E 0C 04
$0840: 4E 0C 04 4E 0C 04 5A 26 F1 AE FF 3B 44 EC 49 B7
$0850: 08 20 E2
```

→ Hinweis: Die OP-Codes können im Manual des Microcontrollers nachgeschlagen werden (Instruction Set Summary).

Adresse	Mnemonic	OC	OP	Kommentare
0 8 2 9	LDA #FF	3		
0 8 2 A	STA $0B	3		
0 8 2 B	STA $47	3		
0 8 2 C	STA $0F	3		
0 8 3 0	LDA #01	3		
0 8 3 1	STA $08	3		
0 8 3 2	LDX #FF	2		
0 8 3 3	MOV #FF $44	4		
0 8 3 4	MOV $0C $04	5		
0 8 3 5	MOV $0C $04	5		
0 8 4 0	MOV $0C $04	5		
0 8 4 1	MOV $0C $04	5		
0 8 4 2	DECX	1		
0 8 4 3	BNE $F1			
	LDX $FF			
	DBNZ	7		
	ROLA			
	STA			
	BRA			

Aus dem MC-Unterricht: Assembler Direktiven
Speicherbereich können in SECTIONs gegliedert werden. Dies birgt den Vorteil in sich, dass die Speicherbereiche (ROM, RAM) sehr schnell für grosse Mengen Variablen gewechselt werden können. Das kann von Vorteil sein, wenn der Code auf einer anderen Plattform (z.B. mit weniger Speicher) portiert wird.

→ Location Counter; LC

Label1:	EQU	$13	//benötigt keinen Speicher
Label2:	EQU	$12345678	

→ Konstanten: SECTION

Label3:	DC.B	$01	//benötigt 1 Byte
Label4:	DC.W	$1234	//1 Word = 2 Bytes; abhängig vom Microcontroller
Label5:	DC.W	$12	//dasselbe hier: 1 Word = 2Bytes
Label6:	DC.L	$11223344	//benötigt 4 Bytes
Label7:	DC	$01, $02, $03	//benötigt 3 Bytes

→ Daten: SECTION

Label8:	DS.B	2	//Reserviert 2 Bytes Speicher
Label9:	DS.W	2	//Reserviert 4 Bytes Speicher
Label10:	DS.L	2	//Reserviert 8 Bytes Speicher

Konkret könnte dies im Assembler-Code wie folgt aussehen:

;main.asm

```
;---------------------------------------------------------------------
; 3. STACK und VARIABLEN
;---------------------------------------------------------------------
VarSec:       SECTION
Cnt:          DS 1

;---------------------------------------------------------------------
; 4. KONSTANTEN
;---------------------------------------------------------------------
ConSec:       SECTION
CntInit:      DC 1

;---------------------------------------------------------------------
; 5. PROGRAMM
;---------------------------------------------------------------------
PrgSec:       SECTION
_Startup:
              LDA CntInit
              STA Cnt
;---------------------------------------------------------------------
```

```
;project.prm
;--------------------------------------------------------------------
SEGMENTS
    Z_RAM                    =  READ_WRITE   0x0080 TO 0x00FF;  //   128 Bytes RAM
    RAM                      =  READ_WRITE   0x0100 TO 0x107F;  //  3968 Bytes RAM
    ROM                      =  READ_ONLY    0x182C TO 0xFFAF;  // 59268 Bytes ROM
    ROM1                     =  READ_ONLY    0x1080 TO 0x17FF;  //  1920 Bytes ROM
    ROM2                     =  READ_ONLY    0xFFC0 TO 0xFFCB;  //    12 Bytes ROM
END
;--------------------------------------------------------------------
PLACEMENT
  DEFAULT_ROM                INTO ROM;
  DEFAULT_RAM                INTO RAM;
  VarSec                     INTO RAM;
  ConSec, PrgSec             INTO ROM;
  _DATA_ZEROPAGE, MY_ZEROPAGE INTO Z_RAM;
END
;--------------------------------------------------------------------
```

Aus dem MC-Unterricht: I/O Ports

→ PTAD $0000: Datenregister Port A

→ PTAPE $0001: PullUp Enable Register für Port A. (Mit Hilfe des PullUp's kann festgelegt werden, ob der Standardwert des Ports auf logisch 1 gesetzt wird).

→ PTASE $0002: Datenregister Port A (Bestimmt, ob der Port "schnell" oder "langsam" umgeschaltet wird. Ein "zu schnelles" Umschalten kann elektromagnetische Störungen verursachen).

→ PTADD $0003: Datenrichtungsregister für Port A (Data Direction; d.h. ob der Port als Ein- oder als Ausgang benutzt wird.