Analyse Maschinencode

1. Versuchen Sie den nachfolgend gegebenen Maschinencode in mnemonischen Code umzusetzen (alle Byteangaben in Hex). Benutzen Sie hierzu die Vorlage "MC_Code.docx". Welche Bedeutung könnte die scheinbar unerklärbare Befehlsfolge in der Mitte des Programms haben?

```
$1960 -------- -------- --A60FB7 0BA680B7
$1970 09A6EFB7 C1B70A48 484848B7 08AEFFA6
$1980 FF4E0602 4E06024E 06024E06 025A26F1
$1990 AEFF4BED B6C149B7 C1B70A48 484848B7
$19A0 0820DA-- -------- -------- --------
```

Start-Adresse (Hex)				Assembler Befehl (Mnemonic)	OpCode (Hex)		Operand1 (Hex)		Operand2 (Hex)		Kommentare
1	9	6	9	LDA #$0F	A	6	0	F			LDA IMM $0F | 0F in Akku laden
1	9	6	B	STA $0B	B	7	0	B			STA DIR $0B | Akku an Addr $0B speichern
1	9	6	D	LDA #$80	A	6	8	0			$80 an Addr $09 speichern
1	9	6	F	STA $09	B	7	0	9			
1	9	7	1	LDA #$EF	A	6	E	F			$EF an Addr $C1 speichern
1	9	7	3	STA $C1	B	7	C	1			
1	9	7	5	STA $0A	B	7	0	A			$EF an Addr $0A speichern
1	9	7	7	LSLA	4	8					LSLA (Logical Shift Left) MSB → Carry
1	9	7	8	LSLA	4	8					4x nach links schieben
1	9	7	9	LSLA	4	8					…
1	9	7	A	LSLA	4	8					Akku Inhalt danach: $F0
1	9	7	B	STA $08	B	7	0	8			Akku ($F0) an Addr $08 Speichern
1	9	7	D	LDX #$FF	A	E	F	F			LDX IMM $FF | $FF in Register X Laden
1	9	7	F	LDA #$FF	A	6	F	F			FF in Akku laden
1	9	8	1	MOV $06 $02	4	E	0	6	0	2	MOV DD (Dest) ← (Src)
1	9	8	4	MOV $06 $02	4	E	0	6	0	2	4x „sinnloser Code“
1	9	8	7	MOV $06 $02	4	E	0	6	0	2	…
1	9	8	A	MOV $06 $02	4	E	0	6	0	2	→ Delay
1	9	8	D	DECX	5	A					Register X Decrement (-1) | Inhalt danach: $FE
1	9	8	E	BNE $F1	2	6	F	1			BNE REL $F1 (zur akt. Addr + $F1) [signed] $F1 = -128 + 113 = -15 → $0F $90 + (-$0F) = $1981
1	9	9	0	LDX #$FF	A	E	F	F			$FF in Register X Laden
1	9	9	2	DBNZA $ED	4	B	E	D			DBNZA INH → REL-Addr. !!! (Decrement Akku and Branch if Not Zero) $ED→ -19 → $1994 - $13 = $1981
1	9	9	4	LDA $C1	B	6	C	1			Wert von Addr. $C1 in Akku Laden | $EF in Akku laden
1	9	9	6	ROLA	4	9					Rotate Left through Carry | Akku Inhalt danach: $DE
1	9	9	7	STA $C1	B	7	C	1			STA DIR $C1 | $DE an Addr. $C1 speichern
1	9	9	9	STA $0A	B	7	0	A			STA DIR $0A | $DE an Addr. $0A speichern
1	9	9	B	LSLA	4	8					LSLA (Logical Shift Left)
1	9	9	C	LSLA	4	8					4x nach links schieben
1	9	9	D	LSLA	4	8					…
1	9	9	E	LSLA	4	8					Akku Inhalt danach: $E0
1	9	9	F	STA $08	B	7	0	8			$E0 an Addr. $08 speichern
1	9	A	1	BRA $DA	2	0	D	A			BRA $DA → REL-Addr. ! $DA = -128 + 90 = -38 → $26 $19A3 - $26 = $197D → Endlos Loop

2. Geben Sie obigen Maschinencode direkt im Debugger Memory Browser ab der Adresse $1964 ein. Benutzen Sie die Debugger-Befehle „reg“ und „go“ um das Programm zu starten und überprüfen Sie Ihre vorausgesagte Funktion.
   done

3. Erstellen Sie ein Assemblerprojekt, das die identische Funktion von 1.1 realisiert.

```
; Subroutinen
;*****************************************************************
Delay:      MACRO
            STA     TempA           ; Akku Inhalt sichern
            LDA     \1              ; Delay Wert als Makro-Parameter, EXT Adressierung
DelLoop:    DECA                    ; Akku dekrementieren bis 0
            BNE     DelLoop
            LDA     TempA           ; Akku Inhalt wieder herstellen
            ENDM

; Start des Benutzer-Codes
;*****************************************************************
userMain:   LDA     #$0F            ; Ports auf Ausgang setzen
            STA     PTFDD
            LDA     #$80
            STA     PTEDD
            LDA     #$EF            ; Bit auf Port F setzen (LEDs Low Active)
            STA     $00C1
            STA     PTFD
            LSLA                    ; Vier mal nach Links schieben
            LSLA
            LSLA
            LSLA
            STA     PTED            ; Auf Port E ausgeben (LED B FR)

Loop:       LDX     #$FF            ; Innerer Schleifenzaehler &
            LDA     #$FF            ; auesserer Schl.zaehler init

Iter:   IF UseMacro = 0
        ; Naive Lösung zum Zeit "verbraten"
            MOV     PTDD,PTBD       ; 4 x dummy Operation
            MOV     PTDD,PTBD
            MOV     PTDD,PTBD
            MOV     PTDD,PTBD
        ELSE
        ; (Etwas) elegantere Lösung zum Zeit verbraten, mit Macro
            Delay   DelVal
        ENDIF


            DECX                    ; X dekrementieren
            BNE     Iter            ; Springe zu Iter solange X != 0

            LDX     #$FF            ; Neu init. innere Schleife
            DBNZA   Iter            ; Dekrementiere A und springe zu Iter solange A > 0


            LDA     $00C1
            ROLA                    ; Bit schieben
            STA     $00C1
            STA     PTFD            ; Ausgabe neue Bitstellung
            LSLA                    ; Akku vier mal nach Links schieben
            LSLA
            LSLA
            LSLA
            STA     PTED            ; Auf Port E ausgeben (LED B FR)

            BRA     Loop            ; Ruecksprung


EndLoop:    BRA     *               ; Endlos-Loop (=Programmende)
```