Unterprogramme, Stack und Parameter

1. Aufruf einer leeren Funktion, ohne Parameter

```
void function(void)
{
   asm NOP;
}

/**
 * Hauptprogramm
 */
void main(void)
{
   function();
   for(;;){}
}
```

Unterprogrammaufruf und Parameterübergabe:

Trace	PC	SP	HX	nächster Befehl C	nächster Befehl Ass
BP:	1AF8	2FF		function1() {	BSR -2
1	1AF6	2FD		asm NOP	NOP
2	1AF7	2FD		}	RTS
3	1AFA	2FF		for(;;) {}	BRA +0

Stack:

		:	SP nach Trace:
	$02FB		
	$02FC		
	$02FD		1, 2
	$02FE	1A	
Bottom	$02FF	FA	BP
		:	

2. Aufruf einer Funktion mit lokalen Variablen, ohne Parameterübergabe

```
void function(void)
{
   unsigned char ucl;
   int il;

   ucl = 1;
   il = (int)ucl;
   ucl++;
}

/**
 * Hauptprogramm
 */
void main(void)
{
   function();
   for(;;){}
}
```

Unterprogrammaufruf und Parameterübergabe:

				nächster Befehl	
Trace	**PC**	**SP**	**HX**	**C**	**Ass**
BP:	1B08	2FF	n.a.	function2() {	BSR -18
1	1AF6	2FD		function2(void) unsigned char uc1 int 1l	AIS #-3
2	1AF8	2FA		uc1 = 1	LDA #0x01
3	1AFA				TSX
4	1AFB		02FB		STA, X
5	1AFC			il = (int)uc1	LDX #0x01
6	1AFE		0201		CLRH
7	1AFF		0001		STHX 2,SP
8	1B02			ucl++	INCA
9	1B03				TSX
10	1B04		2FB		STA, X
11	1B05			}	AIS #3
12	1B07	2FD			RTS 0
13	1B0A	2FF		for(;;) {}	BRA +0
14					

Stack:

		:	SP nach Trace:
	$02F9	uu	
	$02FA	uu	2
ucl	$02FB	~~01~~ 02	
il-H	$02FC	00	
il-L	$02FD	01	1, 12
	$02FE	1B	
Bottom	$02FF	0A	BP, 13

3. Aufruf einer Funktion mit lokalen Variablen, mit Parametern und Rückgabe

```
char function (unsigned char ucVal, unsigned char *ucRef)
{
   unsigned char ucl1;
   ucl1 = ucVal;
   ucl1++;
   (*ucRef)++; /* Achtung: Ohne Klammern, wird der Pointer inkrementiert */
   return ucl1;
}

/**
 * Hauptprogramm
 */
void main(void)
{
   unsigned char uc1=0, uc2=0, uc3;
   uc3 = function(uc1, &uc2);
   uc3++;
   for(;;){}
}
```

Unterprogrammaufruf und Parameterübergabe:

				nächster Befehl	
Trace	**PC**	**SP**	**HX**	**C**	**Ass**
BP:	1B02	02FF	n.a.	main()	AIS #-3
1	1B04	02FC	n.a.	uc1=0	TSX
2	1B05		02FD		CLR 2,X
3	1B07			uc2=0, uc3	CLR, X
4	1B08			uc3 = function3(..	CLR A
5	1B09				BSR -19
6	1AF6	02FA		char function3(...	PSH H
7	1AF7	02F9		ucl1 = ucVal	STA 1, SP
8	1AFA			ucl1++	INC 1, SP
9	1AFD			(*ucRef)++	INC, X
10	1AFE			return ucl1(;;) {}	TSX
11	1AFF		02FA		LDA, X
12	1B00			}	PUL H
13	1B01	02FA	01FA		RTS
14	1B0B	02FC		uc3 = function3 (...	TSX
15	1B0C		02FD		STA 1,X
16	1B0E			uc3++	INC 1,X
17	1B10			for (;;) {}	BRA +0

Stack:

		:	SP nach Trace:
	$02F9	uu	7
ucl1	$02FA	~~02~~ ~~00~~ 01	6, 13
	$02FB	1B	
	$02FC	0B	1, 14
uc2	$02FD	~~00~~ 01	
uc3	$02FE	~~01~~ 02	
uc1 Bottom	$02FF	00	BP

Kontrollfragen:

1. Wie werden die Parameter ucVal und ucRef beim Funktionsaufruf übergeben?
   ucVal wird via Akku (Register) übergeben
   ucRef wird via Stack übergeben

2. Wie heissen diese Arten der Parameterübergabe?
   ucVal: call-by-value
   ucRef: call-by-reference

3. An welchen Adressen liegen die Variablen uc1, uc2, uc3 und ucl1? Warum?
   uc1: $02FF
   uc2: $02FD
   uc3: $02FE
   ucl1: $02FA
   Reihenfolge wie sie deklariert / übergeben wurden.

4. Wie wird das Resultat der Funktion ans Hauptprogramm übergeben?
   Via Akku

5. Wofür werden im Programm die Befehle PSH H und PUL H verwendet?
   Speicher auf dem Stack reservieren / freigeben

6. Welcher andere Befehl wäre möglich?
   PSHH → AIS -1
   PULH → AIS +1
   (AIS: Add Immediate Value (Signed) to Stack Pointer)

7. Warum wird hier PSH H und PUL H verwendet?
   Es wird weniger Programm-Speicher benötigt.
   PSHH / PULH sind 1-Byte Befehle, AIS braucht noch einen zweiten Operanden → 2-Byte