Kontrollstrukturen (Zusammenfassung):

IF a<b THEN aktion1 ELSE aktion2

```
          CMP     a,b
          BGE     aktion2
            •
            •               Ausführung
            •               von aktion1
          BRA     EXIT
aktion2     •
            •               Ausführung
            •               von aktion2
EXIT
```

WHILE a>b DO aktion

```
REPEAT    CMP     a,b
        BLE     EXIT
          •
          •               Ausführung
          •               von aktion
        BRA     REPEAT
EXIT
```

FOR K = I TO J DO aktion

```
          LDB     #i
REPEAT
            •
            •             Ausführung
            •             von aktion
          INCB
          CMPB    #(j+1)
          BNE     REPEAT
EXIT
```

REPEAT aktion UNTIL a=b

```
REPEAT
            •
            •             Ausführung
            •             von aktion
        CMP     a,b
        BNE     REPEAT
EXIT
```

Die Instruktion : LEA (Register) - Load Effective Address

- unterstützt arithmetische Operationen auf den Adreßregistern X,Y,S,U

LEAX, LEAY, LEAS, LEAU lädt nicht den Operanden, auf den die Adresse zeigt, sondern die Adresse selbst !

Beisp.:

•	LEAX	1,X	Increment X
•	LEAY	-1,Y	Decrement Y
•	LEAU	$ABCD,U	Addiere $ABCD zum U-SP
•	LEAX	0,PC	äqu. zu TFR PC,X

- unterstützt positionsunabhängigen Code

Beisp.:

Addr.	OPCODE		OP-Addr.	Mnemonic		
0100	30	8D	0109	START	LEAX	TABLE,PCR
0104	A6	80		LOOP	LDA	,X+
0106					.	
					.	
020D				Table	FCC	/table of whatever/

Der Anfang von “Table“ hat einen Versatz von $10D vom START.
Der Assembler berechnet die Distanz $109, da er den Versatz vom um 4 incrementierten PC berücksichtigt. Durch die Asemblerdirektive “PCR“ wird der Befehl “LEAX TABLE,PCR“ vom Assembler in “LEAX offset,PC “d.h. 30 8D 01 09 übersetzt.

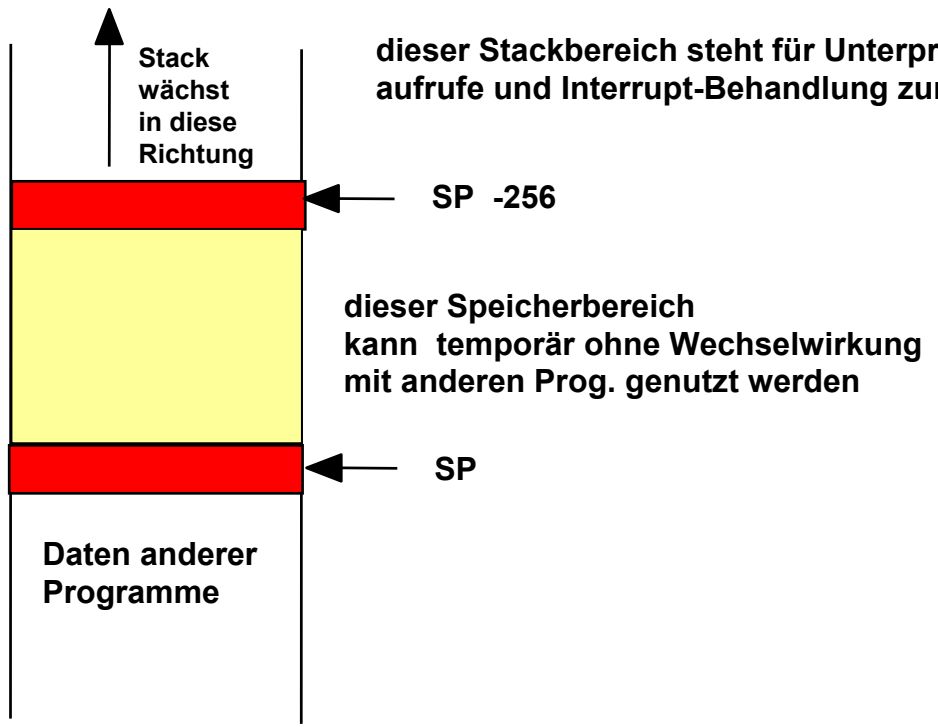
Nutzung des System Stacks als temporärer Speicher

Statische Zuordnung: **(static allocation)**	**Speicher wird fest reserviert, z.B. durch die Assemblerdirektiven : RMB, FCC, FCB, etc. Dieser Speicher kann während der Programmlaufzeit nicht (ohne Gefahr) anderweitig genutzt werden. Auch wenn das entpr. Programm nicht aktiv ist, ist es gefährlich, diesen Speicherbereich zu benutzen, da ein aktivieren des Programms die dort abgelegten Daten anderer Programme überschreibt.**
Dyn. Zuordnung: **(dynamic allocation)**	**Speicher wird während der Laufzeit genutzt und dann wieder freigegeben. Die Nutzung zertört keine Daten von anderen Programmen.**

Stack als temporärer, dyn. zugeordneter Speicher:

LEAS -256,S reserviert den Speicher

LEAS 256,S gibt den Speicher frei

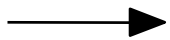


Kontrollstrukturen: Case (Switch) Statement

```
CASE  k (k ≤ N)

I =0        aktion0
I =1        aktion1
I =2        aktion2
I =3        aktion3
            •
            •

I=N         aktionN
```



Assembler-Repräsentation

```
aktion0  FDB    hh
                ll
aktion1  FDB    hh
                ll
aktion2  FDB    hh
                ll
aktion3  FDB    hh
                ll
```

B enthält den CASE Index : x

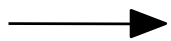
```
                CMPB    #N              gültige Eingabe ?
                BHI     EXEPTION
                ASLB                    Berechng .d. Offsets in die Tabelle (ein Eintrag:2Byte)
                LDX     #aktion0
                ABX                     Addiere Offset in B zu X
                JMP     [  ,X]          Springe indirekt zum Anfang des Progr.-Teils “aktion3“
EXEPTION
```

Kontrollstrukturen:

```
CASE  k (k ≤ N)

I =0        aktion0
I =1        aktion1
I =2        aktion2
I =3        aktion3
            •
            •

I =N        aktionN
```



Assembler-Repräsentation

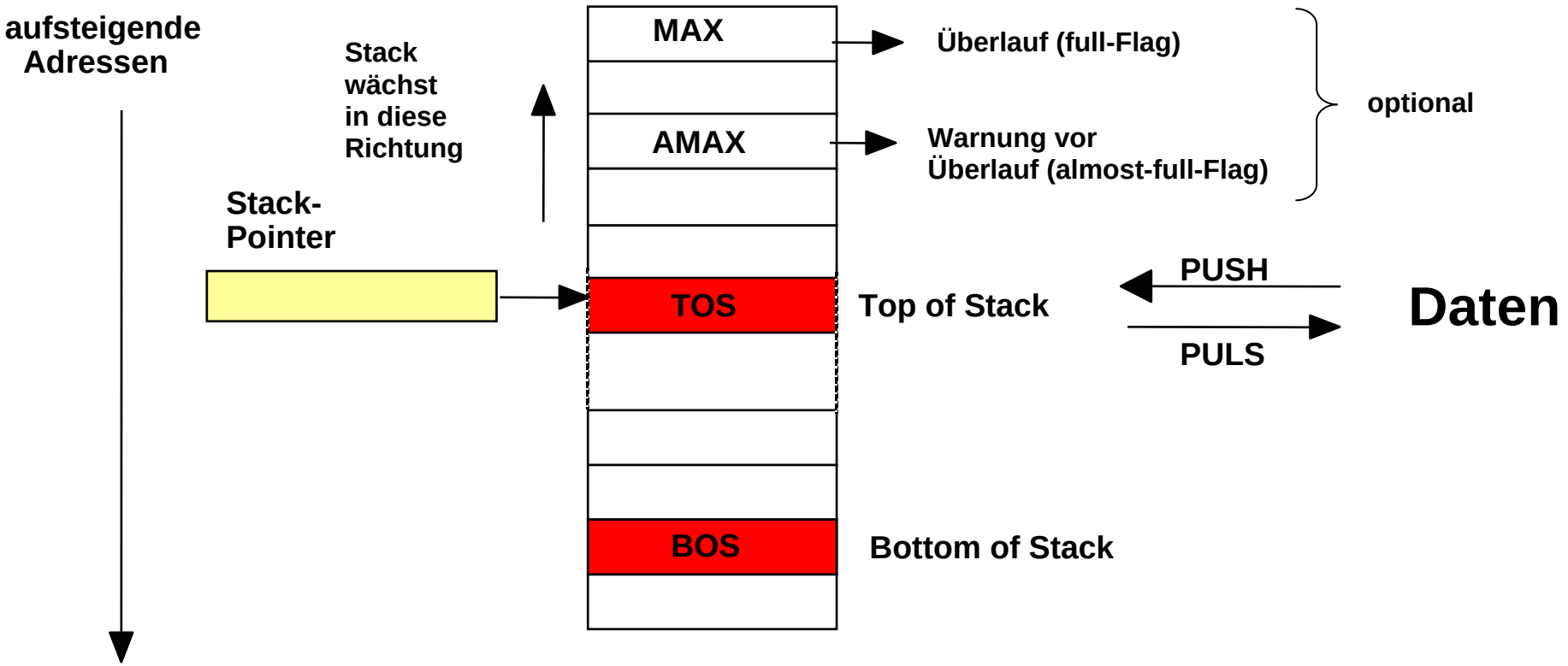
```
aktion0  FDB     hh
                 ll
aktion1  FDB     hh
                 ll
aktion2  FDB     hh
                 ll
aktion3  FDB     hh
                 ll
```

B enthält den CASE Index : x

```
                CMPB    #N              gültige Eingabe ?
                BHI     EXEPTION
                ASLB                    Berechng .d. Offsets in die Tabelle (ein Eintrag:2Byte)
                LEAX    aktion0 ,PCR    Lade Indexregister mit Anfang d. Tabelle relativ zum PC
                JMP      [B  ,X]        Springe indirekt zum Anfang des Progr.-Teils “aktion3“
                                        (B wird Acc.Offset  automatisch zu X addiert)
EXEPTION
```

Hier wird die Tabelle relativ zum Programmzähler adressiert. Außerdem wird beim Sprung der automatische Accumulator Index ausgenutzt.

Stack (Stapel, Keller, LIFO-Queue)



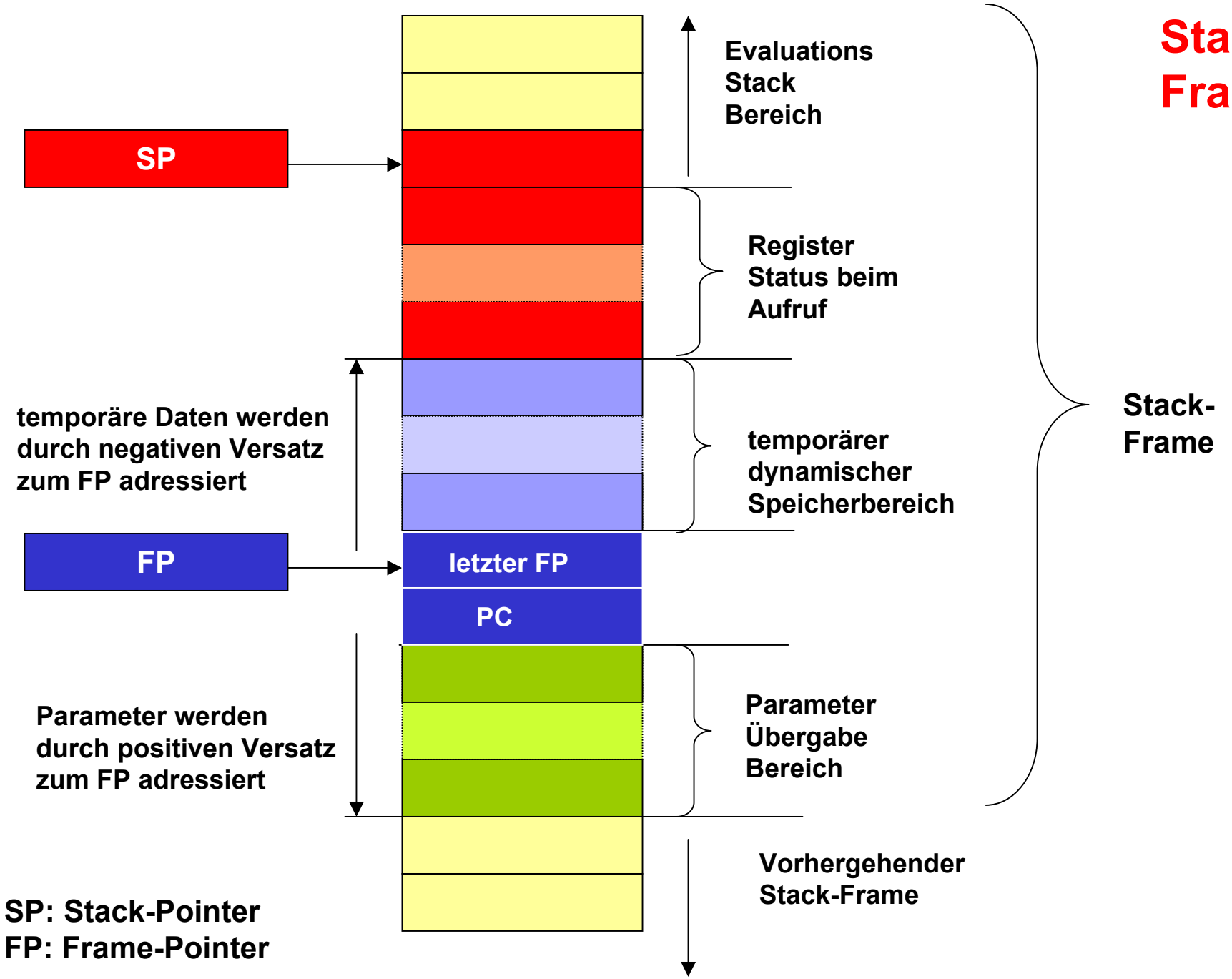
PUSH:

SP=MAX: Exception ‘full‘
SP← SP-1
Mem[SP]← Register
SP >AMAX: Signal ‘almost full‘

PULS:

SP=BOS: Exception ‘empty‘
Register← Mem[SP]
SP← SP+1

Stack-Frame



SP: Stack-Pointer
FP: Frame-Pointer