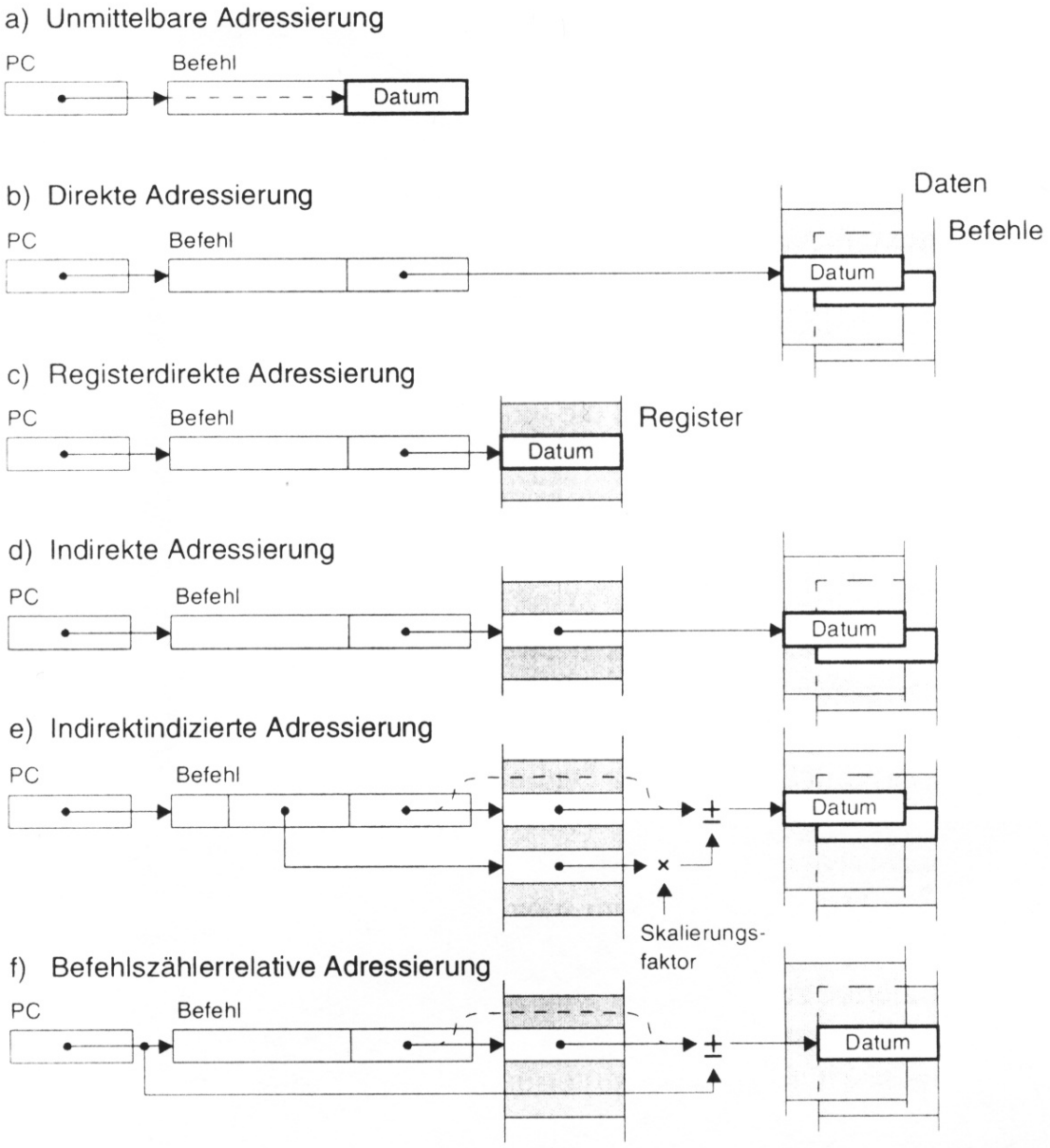
Adressierung und Befehlsfolgen (12)

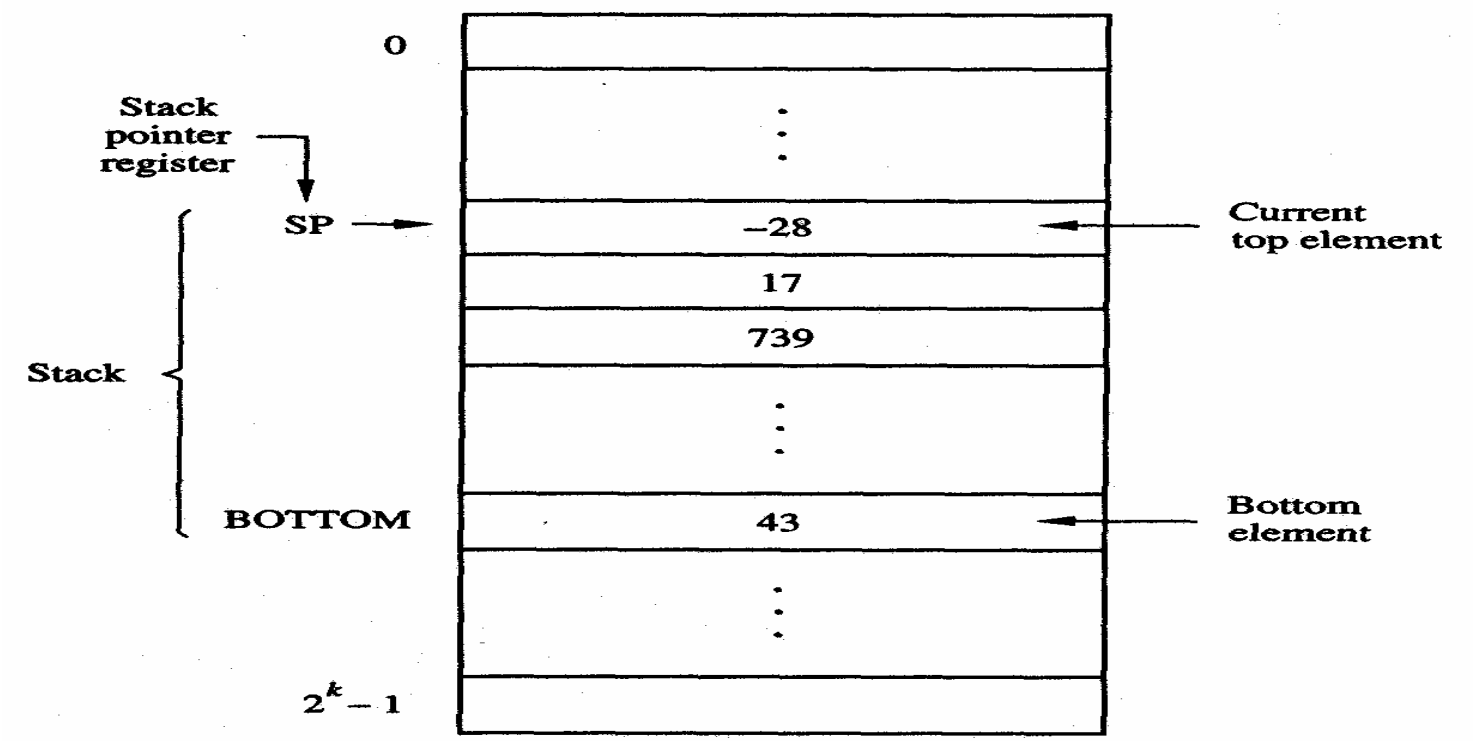


Adressierung und Befehlsfolgen (13)

1.5 Stack (Stapel, Keller)

Definition:
Liste von Datenelementen (meistens Worte oder Bytes), welche nur von einem Ende der Liste (top) bedient werden kann (*push* - *und pop* - Operation ---> *pushdown* bzw. *LIFO* - Prinzip).

Dateispeicher organisiert als Stack:



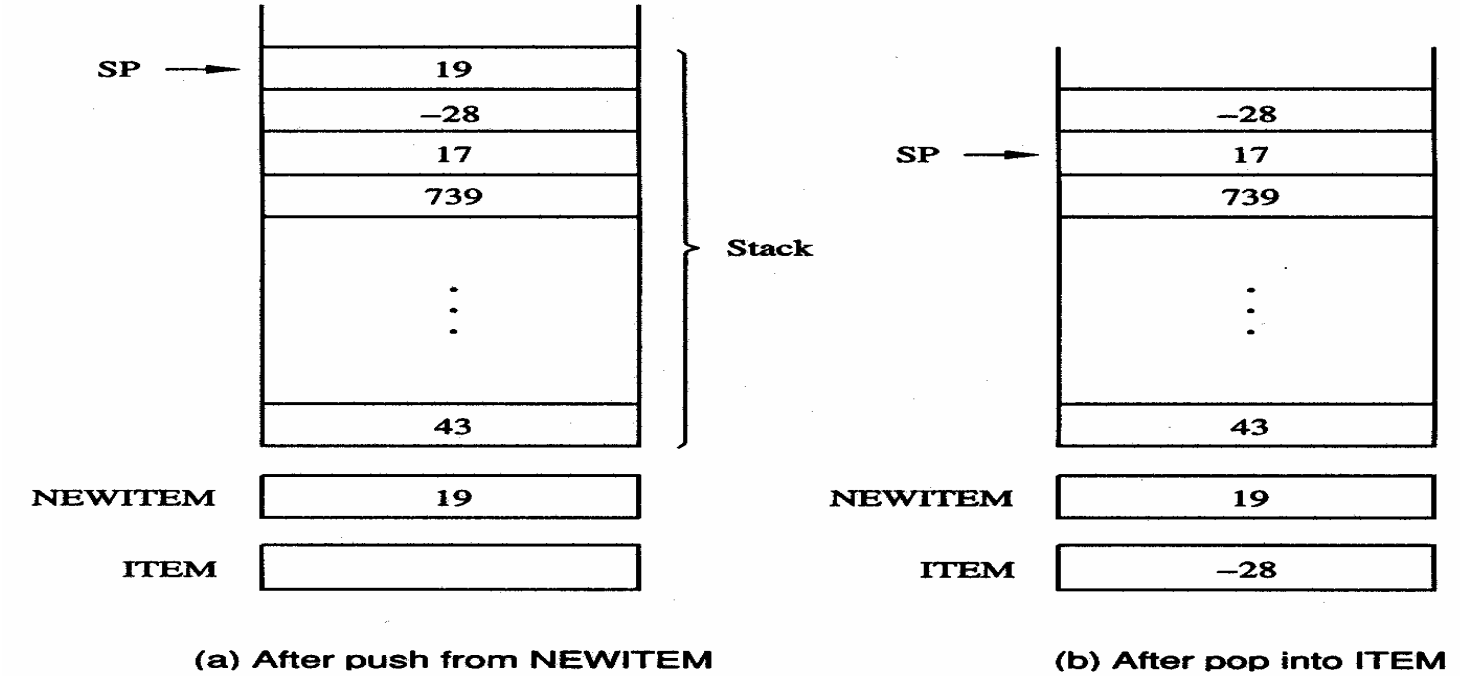
Adressierung und Befehlsfolgen (14)

stack pointer (SP):
CPU-Register, welches die Adresse des momentan oben liegenden Stapelementes enthält.

push - Operation:
Decrement SP
Move NEWITEM, SP

pop- Operation:
Move SP, ITEM
Increment SP

Beispiel:



Adressierung und Befehlsfolgen (15)

Sichere Stack-Operationen:

SAFEPOP	Compare Branch > 0	#2000,SP EMPTYERROR	Check to see if the stack pointer contains an address value greater than 2000. If it does, the stack is empty. Branch to the routine EMPTYERROR for appropriate action.
	Move	(SP)+, ITEM	Otherwise, pop the top of the stack into memory location ITEM.

(a) Routine for a safe pop operation

SAFEPUSH	Compare Branch ≤ 0	#1500,SP FULLERROR	Check to see if the stack pointer contains an address value equal to or less than 1500. If it does, the stack is full. Branch to the routine FULLERROR for appropriate action.
	Move	NEWITEM,−(SP)	Otherwise, push the element in memory location NEWITEM onto the stack.

(b) Routine for a safe push operation

Adressierung und Befehlsfolgen (16)

1.6 Queue (Schlange)

Definition:

Liste von Datenelementen (meistens Worte oder Bytes), welche von beiden Enden der Liste (top) bedient werden kann (push von hinten (unten), pop von vorne (oben) ---> *FIFO* - Prinzip).

Unterschiede zum Stack:

- Beide Enden der Queue bewegen sich als Folge von I/O - Operationen
- Es werden 2 pointer (Zeiger) zur Markierung der Enden benötigt
- Eine Queue kann kontinuierlich durch den Speicher wandern

Adressierung und Befehlsfolgen (17)

1.7 Subroutines (Unterprogramme)

Call_Subroutine ist ein spezieller unbedingter Verzweigungsbefehl bestehend aus:

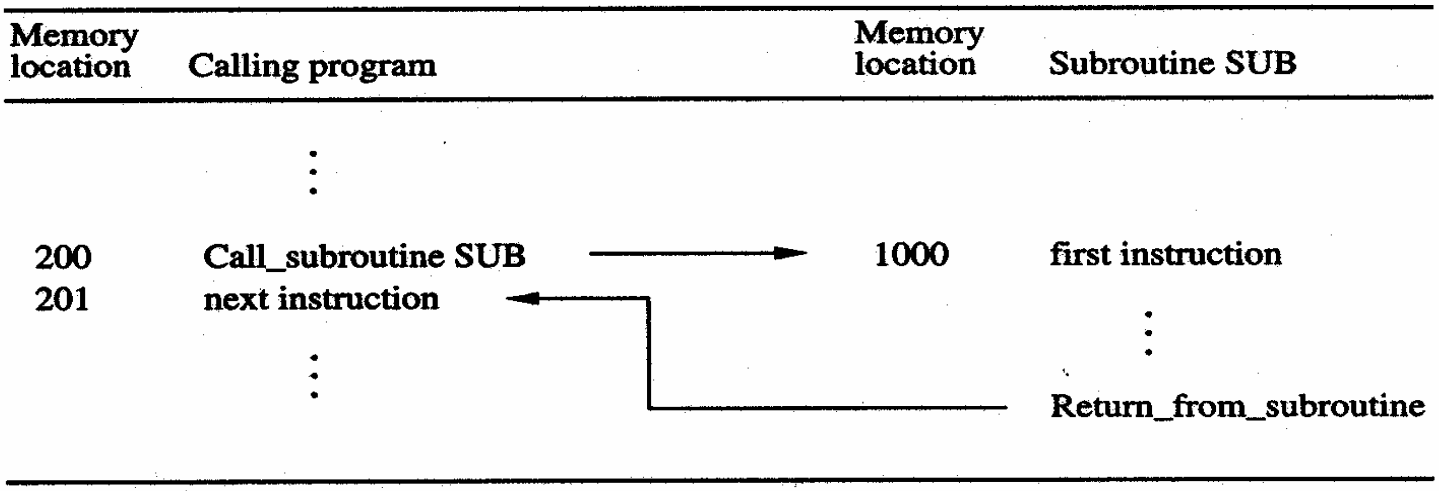
- Speichere den Inhalt des PC (program counter) in das *link register*
- Verzweige zu der im Befehl spezifizierten Zieladresse

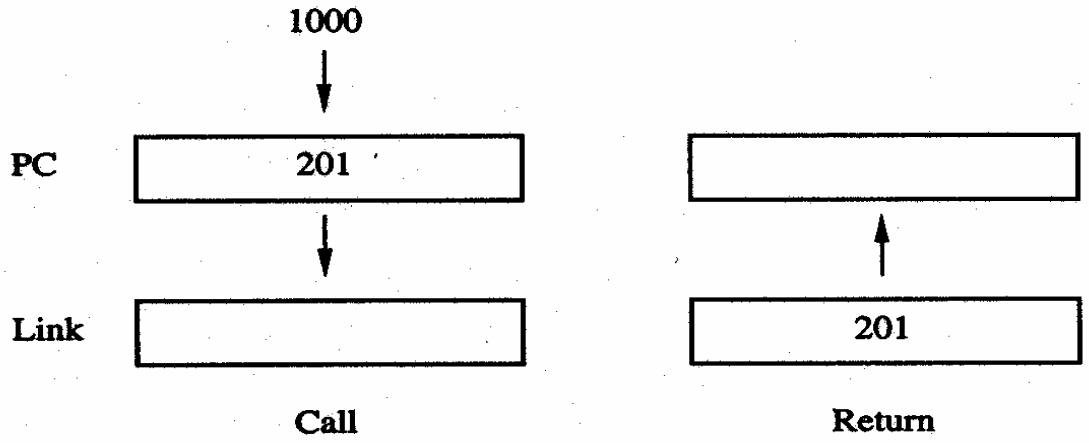
Return_from_Subroutine besteht aus:

- Verzweige zu der im link register enthaltenen Adresse

Adressierung und Befehlsfolgen (18)

Unterprogrammaufruf (subroutine linkage) mit Hilfe des link registers





Adressierung und Befehlsfolgen (19)

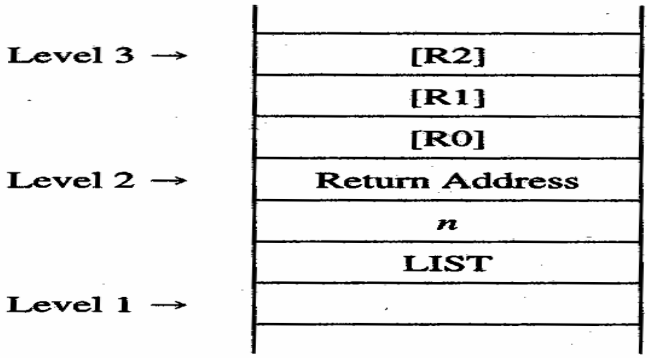
1.7.1 Parameterübertragung

Beispiel: Das Programm „Addiere n Zahlen“ als aufrufbare subroutine

```
Assume top of stack is at level 1 below.

                Move            #LIST,-(SP)     Push parameters onto stack.
                Move            N,-(SP)
                Call            LISTADD         Call subroutine
                                                    (top of stack at level 2).
                Move            1(SP),SUM       Save result.
                Add             #2,SP           Restore top stack
                                                    (top of stack at level 1).
                .
                .
                .
LISTADD         Move_multiple   R0-R2,-(SP)     Save registers
                                                    (top of stack at 3).
                Move            4(SP),R1        Initialize counter to n.
                Move            5(SP),R2        Initialize pointer to the list.
                Clear           R0              Initialize sum to 0.
LOOP            Add             (R2)+,R0        Add entry from list.
                Decrement       R1
                Branch>0        LOOP
                Move            R0,5(SP)        Put result on the stack.
                Move_multiple   (SP)+,R0-R2     Restore registers.
                Return
```

(a) Calling program and subroutine



(b) Top of stack at various times