1. Beschreibung des Routinenprotokolls für MAVL

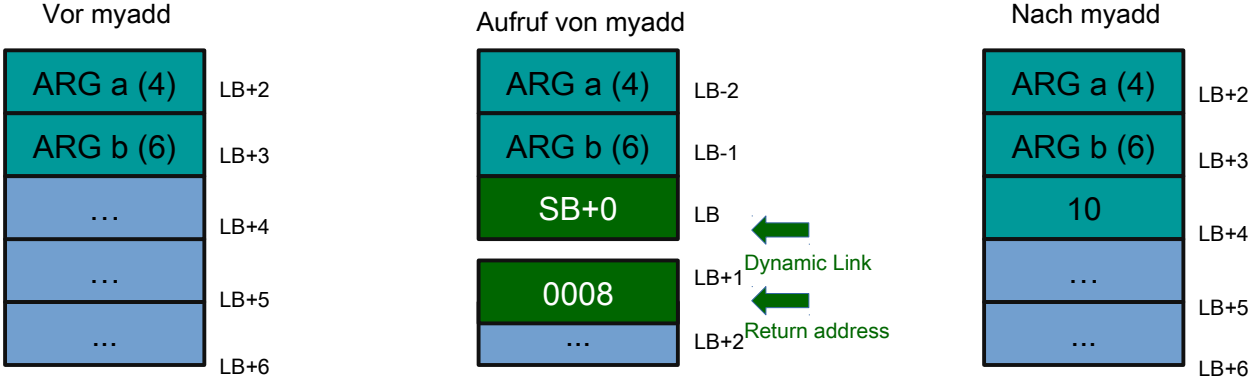
Kompilierte MAVL-Programme werden auf einer abstrakten, virtuellen Machine, welche auf der TAM basiert, ausgeführt. In MAVL werden Argumente und Rückgabewerte bei Funktionsaufrufen mit Hilfe des Stacks übergeben. Die TAM bringt dafür zwei relevante Instruktionen mit:

- **CALL (*CB*) reg[addr]**
  Ruft die Funktion an Adresse *reg+addr* auf und legt einen neuen Stackframe an. Da es in MAVL keine verschachtelten Funktionen gibt, entfällt im Vergleich zur TAM die Angabe des Static Links; daher ist das *n*-Feld der Instruktion per Konvention immer *CB* und wird ignoriert.
- **RETURN (*n*) d**
  Sichert n Worte als Ergebnis vom Stack, entfernt den aktuellen Frame sowie d Parameter, legt das Ergebnis oben auf dem Stack ab und setzt die Ausführung nach der Aufrufstelle fort.

Beispiel:

```
function int myadd(int a, int b) {
    val int x = a + b;
    return x;
}

function void main() {
    val int r = myadd(4, 6);
    val int t = myadd(5, 8);
    myadd(r, t);
}
```

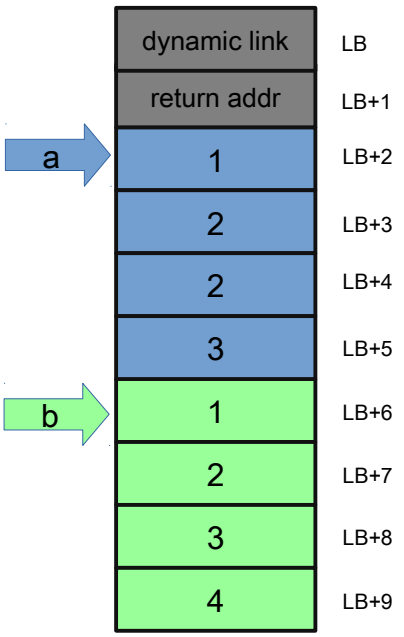
```
0000:  CALL  (CB)  6[CB]    main

# function INT myadd(INT, INT)

0001:  LOAD  (1)   -2[LB]
0002:  LOAD  (1)   -1[LB]
0003:  add
0004:  LOAD  (1)   2[LB]
0005:  RETURN(1)   2
# function VOID main()
0006:  LOADL       4
0007:  LOADL       6
0008:  CALL  (CB)  1[CB]    myadd
0009:  LOADL       5
000A:  LOADL       8
000B:  CALL  (CB)  1[CB]    myadd
000C:  LOAD  (1)   2[LB]
000D:  LOAD  (1)   3[LB]
000E:  CALL  (CB)  1[CB]    myadd
000F:  POP   (0)   1
0010:  HALT
Done.
```



Durch den Call zu myadd werden der alte LB und die Rücksprungadresse auf dem Stack gesichert, einen Static Link gibt es im Gegensatz zur TAM nicht. LB wird angepasst und die Argumente sind nun über LB-1 und LB-2 verfügbar.

2. Speicherlayout der Matrizen und Vektoren

Matrizen und Vektoren werden linear auf dem Stack gespeichert. Die Dimensionen sind statisch zum Zeitpunkt der Kompilierung bekannt und müssen nicht explizit gespeichert werden.




```
function void foo(){
    val matrix<int>[2][2] a = [[1,2],[2,3]];
    val matrix<int>[2][2] b = [[1,2],[3,4]];
}
```