15.6.2012

weiter zu 3.3

Erweiterung der Prozesorgrundstruktur

3.3.2 Erweiterte Operandenadressierung

- Segmentierte Adressierung

Segmentierung (F3_90)

Teilprogramme teilen sich den gesamten Adressraum (oder den darin implementierten physischen Speicher)

jedem TPi ist ein Operandenbereich i zugeordnet.
Adressieren im Operandenbereich über Basisadresse (Anfangsadresse) und Offset (Abstand zur Anfangsadresse)

BA:=Basis-OA+Offset-OA (durchgeführt unmittelbar vor dem Speicherzugriff durch Spezialaddierer (unvorzeichenbehaftete ganzahlige Binärzählen) (entspricht auch der Darstellung in F3_80 links)

analog Befehlsadresse (TRi haben auch getrennte Befehlscodebereiche, Adressierung über Basis und Offset, Unterschied: Offset wird entsprechend ENBA (Erzeugen nächste Befehlsadresse) am Ende des Befehls verändert, Basis bleibt dabei konstant.

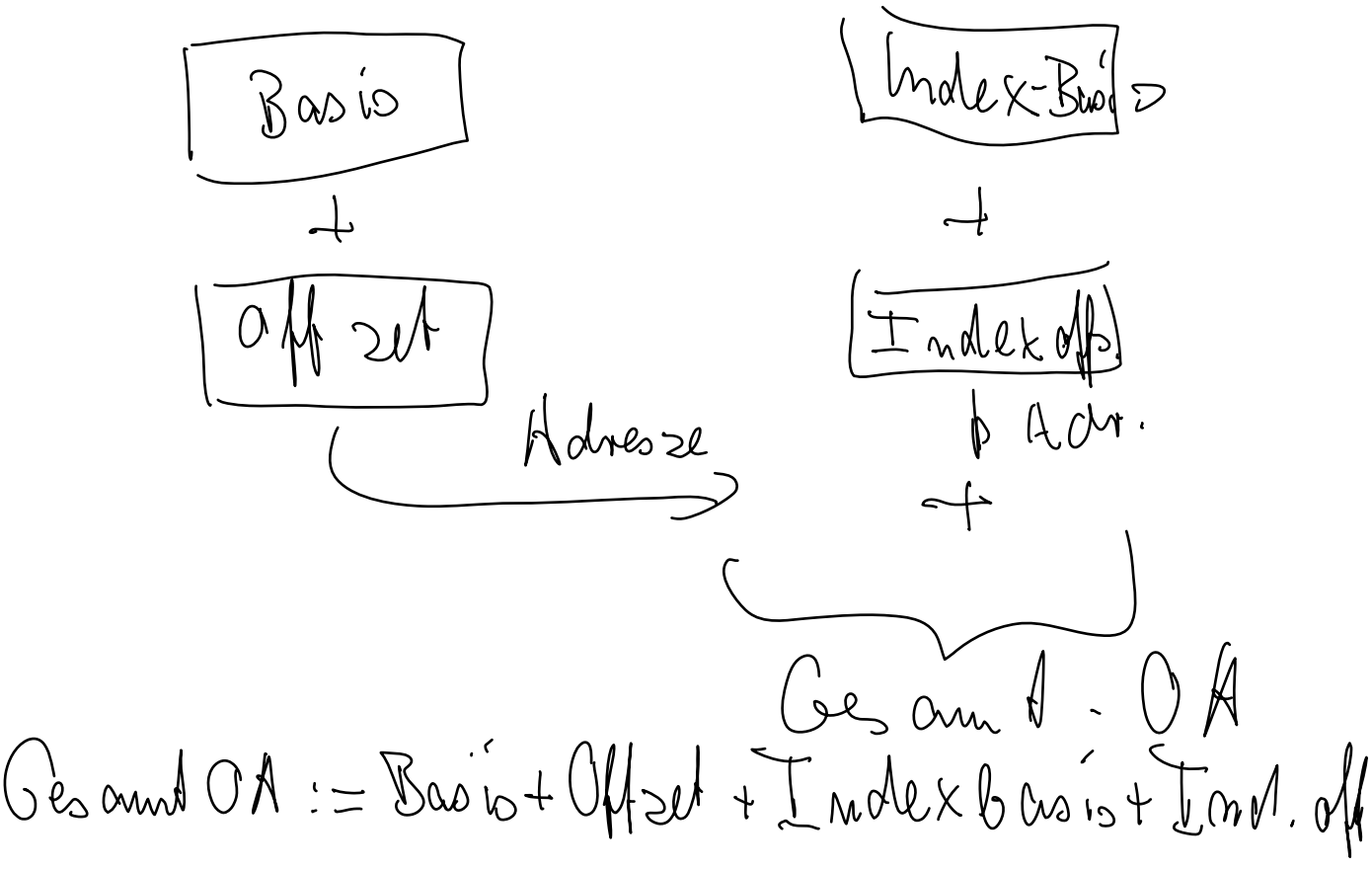
Indexadressierung für Datenstrukturen mit mehreren gleichen Elementen (z.B. Vektoren, Matritzen, Textfelder, Bilder usw.) und Zugriff auf einander folgende Elemente

wie Basisadr., Unterschied nach Zugriff auf einen Opearnden: Veränderung des Offsets um +n (n: Länge der Elemente im Speicher, +: auf El. i folgt El i+1) bzw. –n (auf El.i folgt El. i-1)

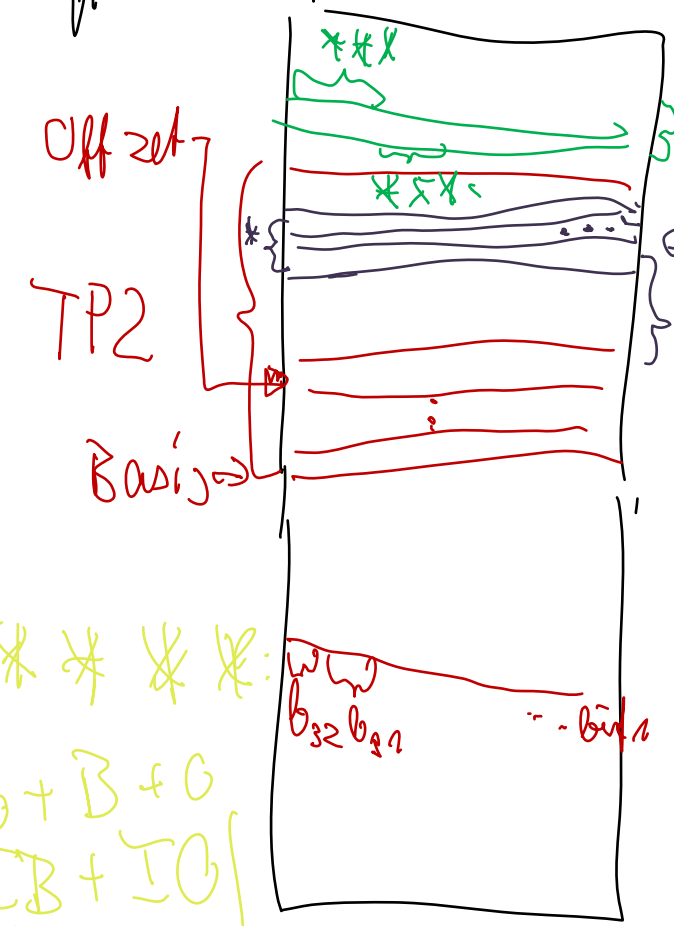
➔ F3_80

Kombination von Adressierungsarten ist typisch:

z.B. Basisadr. + Indexadr.



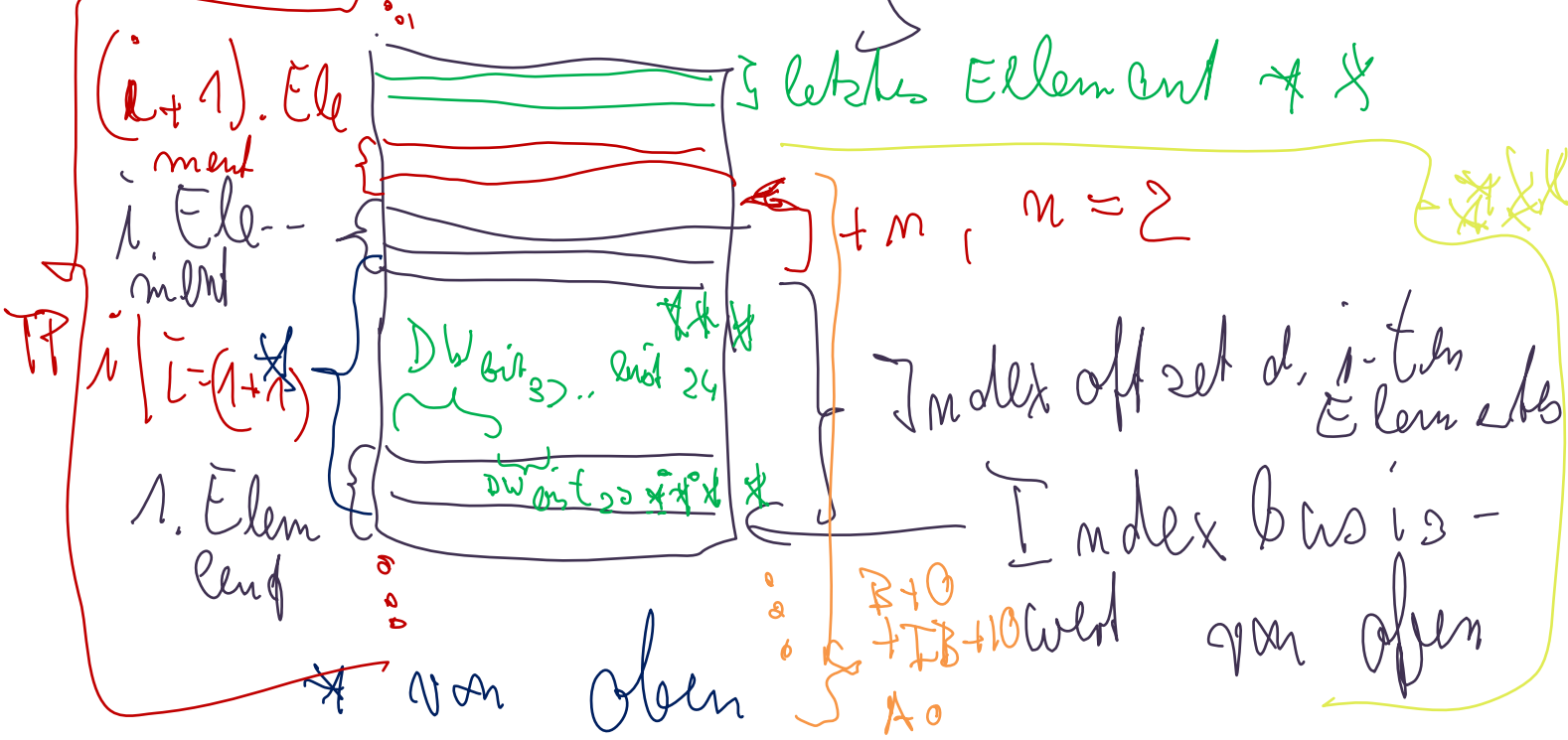
Bsp Vektor, Element 2 DW



A0 + B + C + IB + IO
letztes El

* Index offset
typ.: bit 31 ... bit 1
hier bit 32 ... bit 1

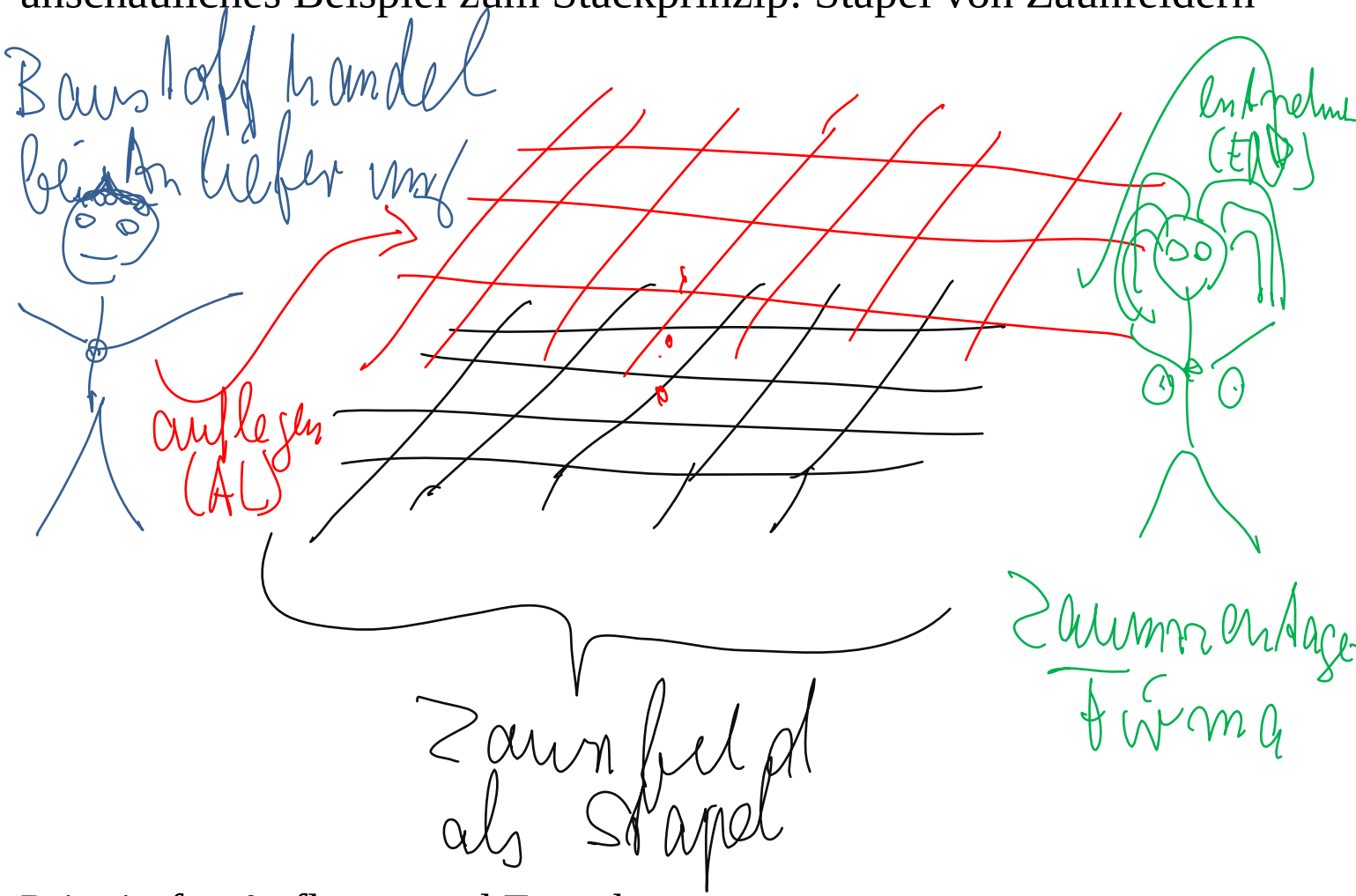
Detail-Darstellung Vektor



Stackadressierung und Stack

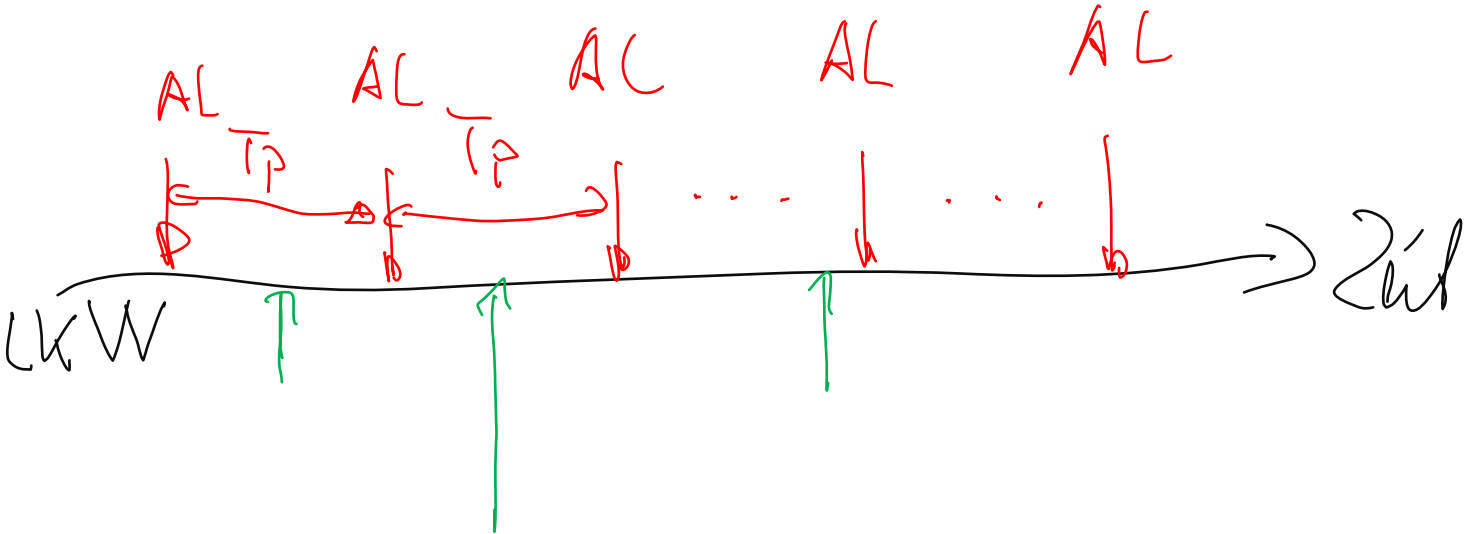
Stack entspricht „Stapel“

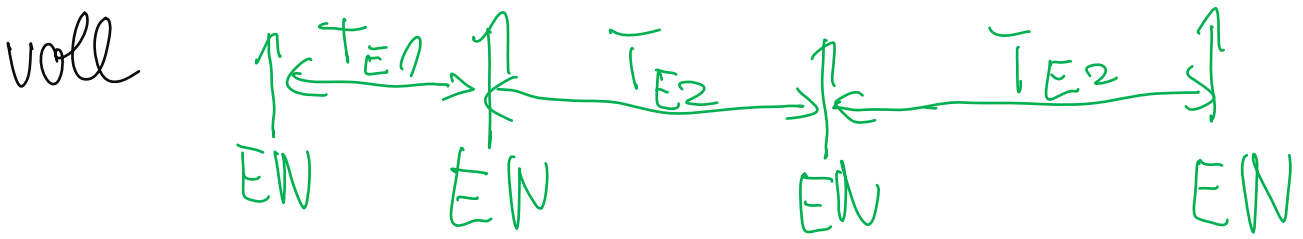
anschauliches Beispiel zum Stackprinzip: Stapel von Zaunfeldern



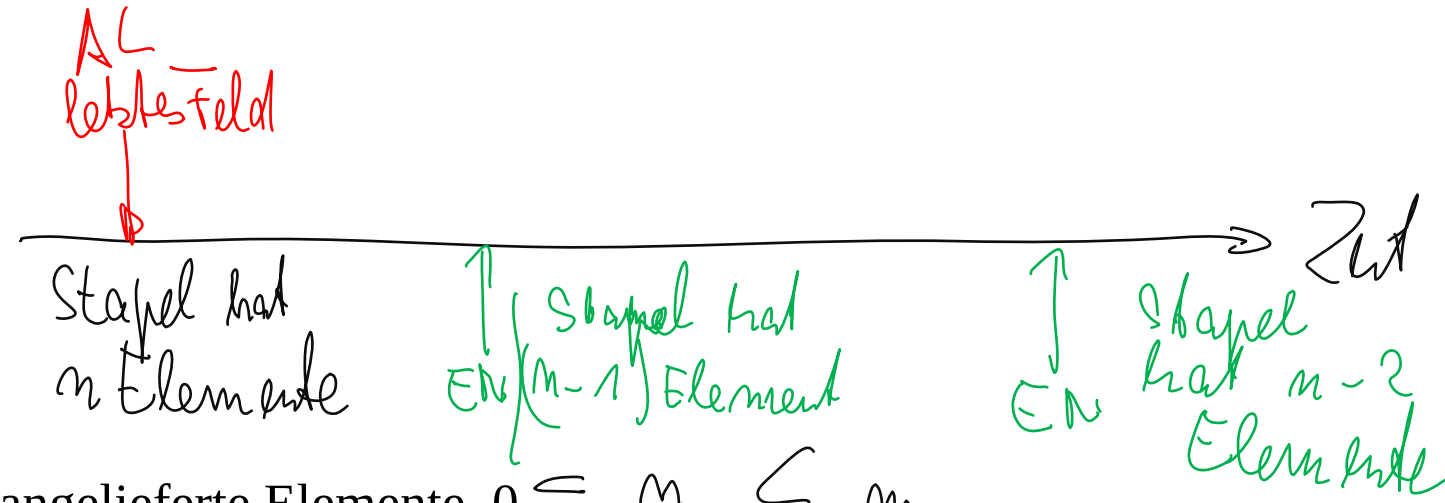
Prinzip für Auflegen und Entnehmen:

Zeitverlauf:





TP etwa const t= f(s,v)



m: angelieferte Elemente, $0 \subseteq m \subseteq m$

Prinzip ist LIFO (Last in First out) siehe Alg. u. Prog.

Elemente-> Datenworte
AL: Speicher schreiben
EN: Speicher lesen