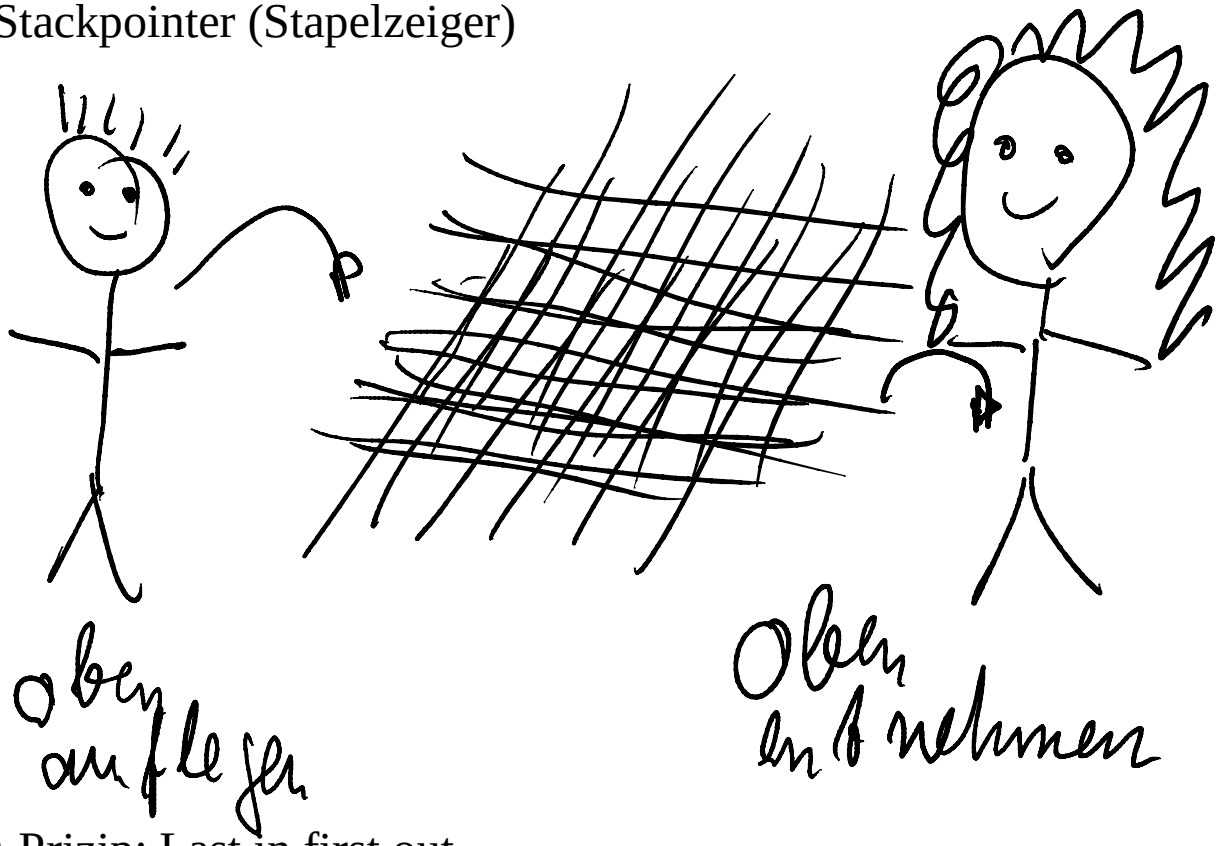
- Stackpointer (Stapelzeiger)



LIFO-Prizip: Last in first out

➔ Prizip eines Stacks im Rechner

Ablauf F3_110: (Bsp)

1. Schreiben: SP:=Sp-1; Schreiben erster Wert
2. Schreiben: SP:=Sp-1; Schreiben 2. Wert

3. Lesen: Lesen 2. Wert; SP:=SP+1
4. Lesen: Lesen 1. Wert; SP:=SP+1

Stackoperationen im Mittel: Leseanzahl = Schreibanzahl

- Mehrere ALE’s parallel in einem Prozessor:

Grund bei gleichen ALE’s: Leistungssteigerung (2 Befehle können gleichzeitig augeführt werden)

Grund bei unterschiedlichen spezialisierten Einheiten: Geringere Logik in den ALE's bei gleichzeitiger Ausführung von Befehlen unterschiedlicher Spezialisierung (z.B Festkommabefehl und Gleitkommabefehl)

- Allgemeiner Befhlsablauf

(Erweiterung zum Ablauf von 3_60)

➔ F3_130

- Minimaler Ablauf
  BL ; BA ; Ebadr (alle E- und A-Operanden im Register)
- Ein mittlerer Ablauf (entspr. 3_60)
  BL ; 1.OL ; BA; Ebadr (1 E-Operand im Speicher, 2. E- und A-Operand in Registern)
- Maximaler Ablauf
  BL : 1.OL ; 2: OL ; BA : A_OS (RS) ; Ebadr

Zeitverbrauch (bei Annahme, dass jede Phase eine Zeiteinheit (ZE) benötigt):

- Min. A. 3 ZE
- Mit. A. 4 ZE
- Max. A. 6 ZE

3.4. Befehlübersicht

➔ Hier Darstellung allgemein, konkrete Befehle in der Übung und den Arbeitsblättern dazu

3.4.1. Datentransferbefehle

- Verändern den Ort von Operanden, verändern diese aber nicht.
- Zumeist Prinzip Kopieren; dh. Ziel wird überschrieben, Quelle bleibt erhalten
- Ziel, Quelle: Register, Speicherplätze; EA-Schnittstellen (evtl. nicht alle Kombinationen möglich)

- Transport (Register, Speicher)
- Stack (Speicher mit OA aus SP (siehe 3.3.)
- EA (Ziel oder Quelle sind EA-Schnittstelle)

3.4.2. Arithmetik- und Logikbefehle
- Operandenmanipulation durch
- Grundrechenarten (Addition, Subtraktion, Multiplikation, Division), koplexere Berechnungen müssen aus den Grundrechenarten zusammmen gesetzt werden (Folge von Befehlen)
  - Vergleich: Subtraktion, Ergebnis wird nicht geschrieben, aber PSR (Flags) werden gesetzt -> mit bedingten Sprüngen in Abhängigkeit der Flags auswertbar
  - Konvertierung: Umwandlung zwischen verschiedenen Zahlenformaten (z.B. binär in BCD)

  - Logische Grundoperationenen:

➔ UND, ODER, EXKLUSIVE_ODER (Antivalenz), NICHT

➔ Bezieht sich immer auf ein ganzes Register oder Speicherplatz (n bit), bit mit dem gleichen Index werden gleichzeitig verküpft und erzeugen je ein Ergebnisbit, wieder mit dem gleichen Index (F3_150) Notwendig: vor der Operation die Positionen der interessierenden bit angleichen

Testbefehle: Einzelbit aus Register oder Speicher in das Zero-Flag negiert -> mit bedingten Sprüngen auswertbar

3.4.3. Schiebe- und Bitmanipulationsbefehle

Schieben:Veränderung von Bitposionen über das ganze Register/Speicherplatz (Bsp F3_160: alle bits werden um zwei verschoben, die letzten zwei entfallen, die vordersten zwei werden durch 0 ergänzt)
- Logische interpretation: Veränderung von Bitpositionen

- Arithmetische Interpretation (Multiplikation mit 2 hoch n (links); Division durch 2 hoch n (rechts)

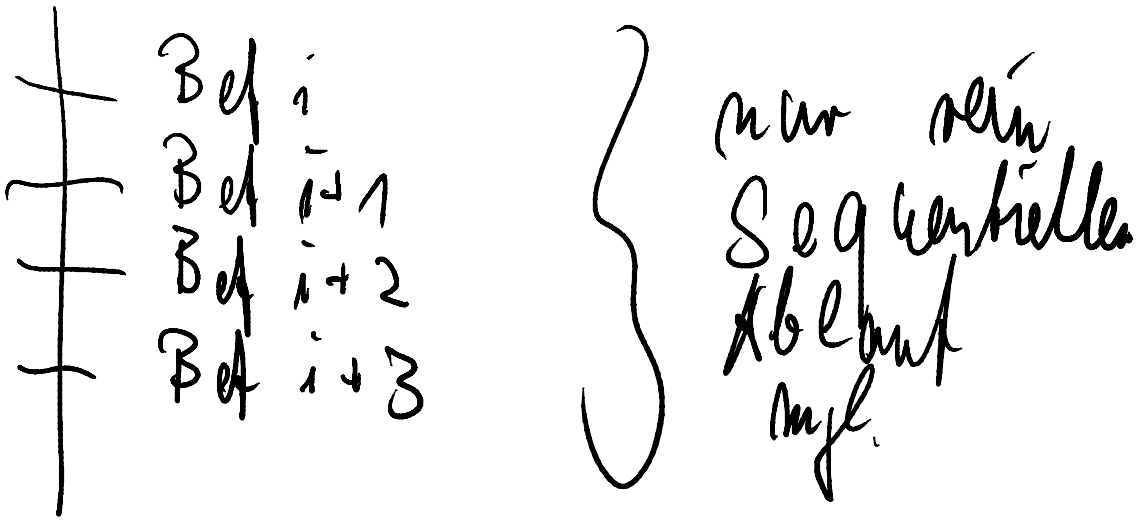
Rotieren: wie Schiben, aber herausgeschobene bit werden auf der anderen Seite wieder herein geschoben.

- Nur logisch interpretierbar, nicht arithmetisch

Bit-Setzen/Rücksetzen (einzelne bit auf 1 bzw. 0 schreiben ohne die anderen bit im Register/Speicherplatz zu verändern)

3.4. Programmtransferbefehle

Bisherige Befehle: Ebadr: Badr:=Badr+1



Für die meisten Algorithmen sind aber (bedinge) Verzeigung und Zusammenführung notwendig (siehe F 3_180 links)

➔ Sprungbefehle

Unbedingter Sprung:

➔ Nach Ausführung Fortsetzung der Befehlabarbeitung immer an einer anderen festen Befhlsadresse (z.B. F3_180 liks unten)

Ablauf:

1. Befehlscode (Sprung) lesen
2. BA:=BA+1 ; Sprungadresse lesen
3. Eadr.: BA:= gelesene Sprungadresse

➔ Nächstes Befehl lesen benutzt gelesene Sprungadresse

Bedingte Sprünge:

- Wie unbedigte Sprünge bei erfüllter Bedingung
- Bei nicht erfüllter Bedingung BA:=BA+1 (d.h. linear sequentiell weiter)
- Bedingungen: PSR-bit (Flags)

Unterprogrammbfehle

Unterprogramm: mehrfach nutzbares Teilprogramm (Grundlage z.B. für Funktionsaufrufe, z.B Sinusberechnung)

Unterschied zu den Sprüngen: Zzusammenführung immer auf den Befehl, der dem Unterprogrammaufruf folgt (F3_180 rechts: grün 1. Aufruf + Rückkehr; blau 2. Aufruf und Rückkehr)

UP-Aufruf (CALL):

1. BL Code CALL
2. BA:=BA+1 ; Sprungadresse lesen
3. BA:=BA+1 ; Schreiben BA in den Stack (mit SP-Manipulation)
4. BA:= gelener Sprungadresse (zeigt auf den 1. Befehl des Unterprogramms)

➔ Wie Sprung, aber Rückkehradresse im Stack zischengespeichert.

UP-Rückkehr (RETURN):

1. BL Code RETURN
2. Lesen vom Stack nach BA (mit SP-Manipulation)

Durch LIFO-Prinzip ist UP-Aufrucf und UP-Rückkehr auch in einem UP möglich. (F3_190)

UP-Befehle können auch bedingt sein (siehe bedingte Sprünge).