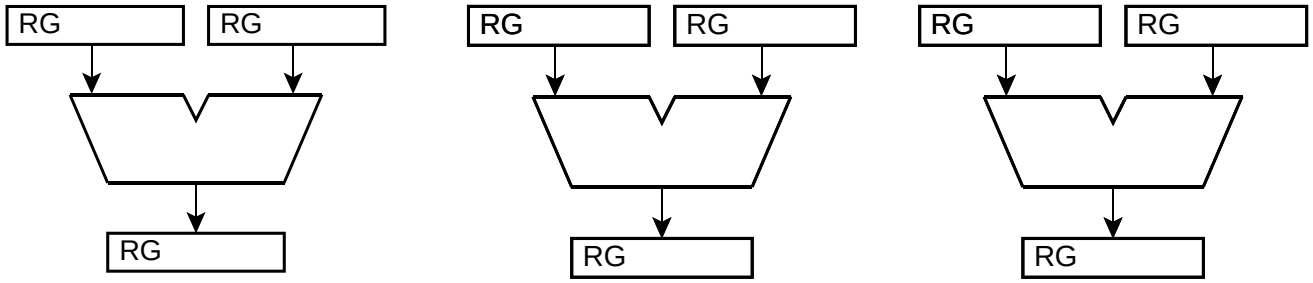
Elementare ReAl Maschinen

Das 1. Verarbeitungswerk Das 2. Verarbeitungswerk Das 3. Verarbeitungswerk



Die Werke -- mit ihren vor- und nachgeschalteten Hardware-Registern (RTL) -- sind so und so vorhanden (gleichgültig, welche Befehlsformate).

Wie kann man sie mit ReAl-Operatoren ansprechen?
Die Register einzeln?
Oder immer einen ganzen Block?

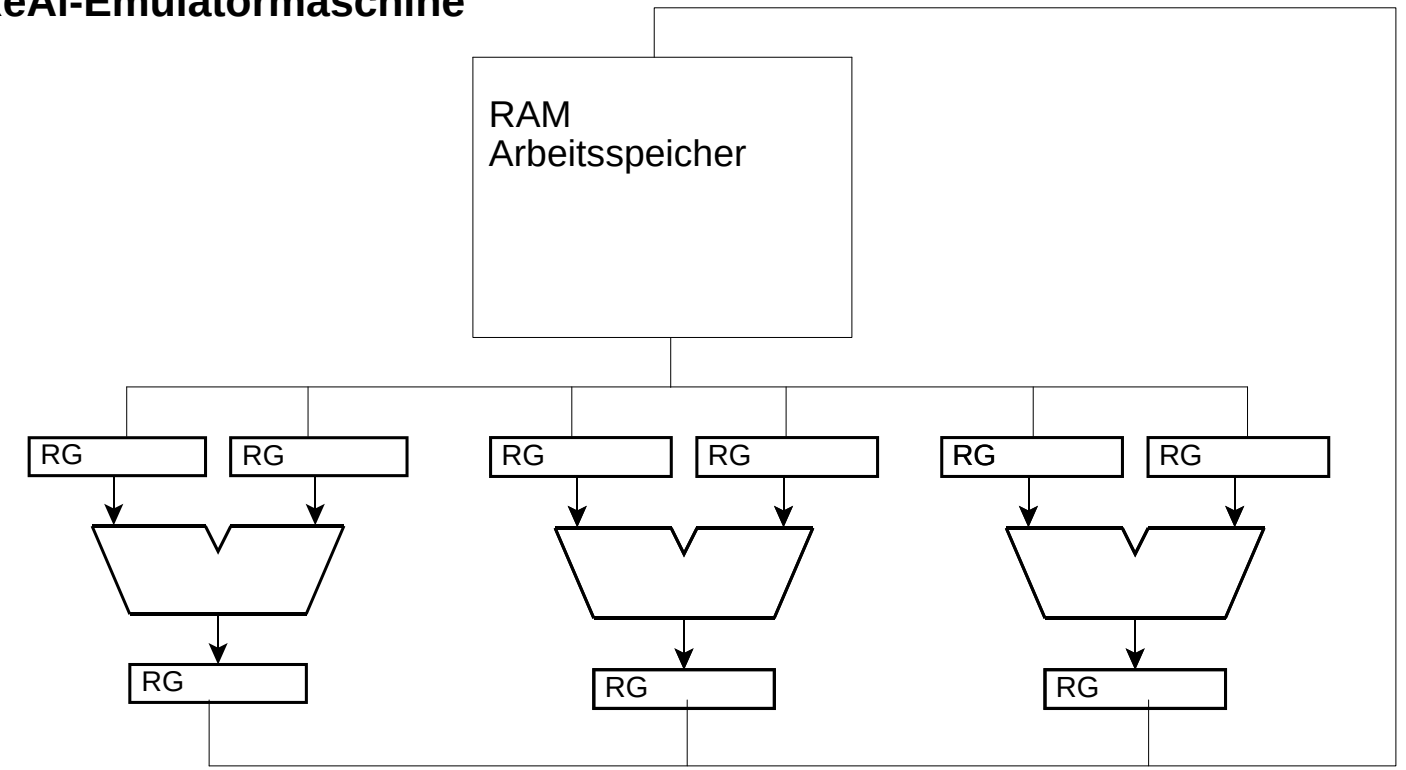
Es gibt reale (physische) Ressourcen und virtuelle. Eine virtuelle Ressource ist praktisch ein Registerabbild im Speicher.

Der Y-Operator bewirkt, daß das Abbild in die Hardware geschafft wird. Wie ist das zu steuern? Das mitgespeicherte Ressourcensteuerwort ist eine Art Mikrobefehl.

Load-Store und Registerbefehle: Im Grunde ist es nichts anderes als ein Speicherbereich, der direkt adressiert wird und der Virtualisierung nicht unterworfen ist. Verarbeitungsbefehle haben nur direkte Operandenadressen. Deshalb gibt es keinenGrund, sich nur auf 3 bis 5 Adreßbits zu beschränken.

7. 3. 2014

Die ReAl-Emulatormaschine



Ein gemeinsamer Arbeitsspeicher. Ggf. Tags. Y-Operator zeigt auf Steuerwort. Dieses wiederum wählt das Operationswerk aus und veranlaßt die Datentransporte.
Ressourcen-Basisadresse, Offset.

Eine Art Mikrobefehl:

Ress.-Offset	HDW-Register	OP
3	3	2

Oder es ist eine Art einfacher Steuerzähler.
Oder die Registeradresse (u. ggf. ein Längencode o. dergl.)

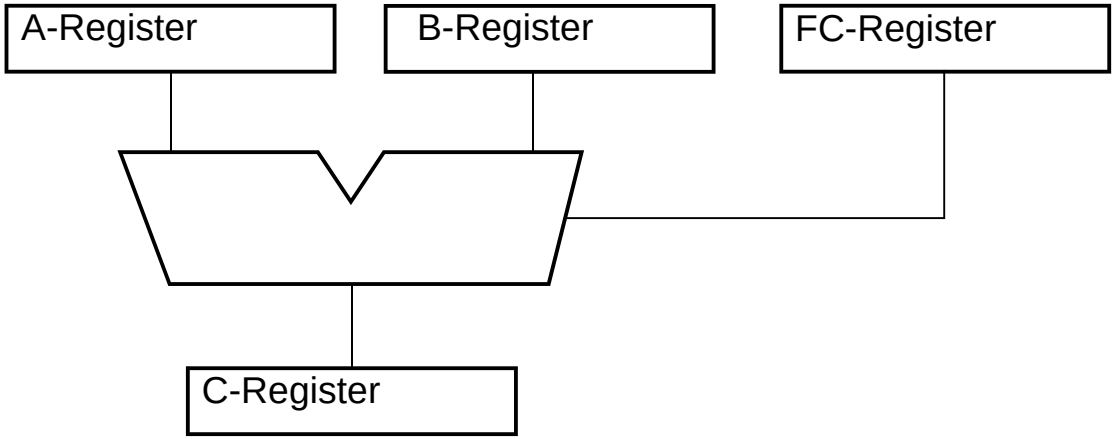
Steuerwort:

Ress.-Typ	Ress.No.	

7. 3. 2014

Das ist eine Hardware-Ressource.
Die Hardware-Register sind
unbedingt erforderlich.

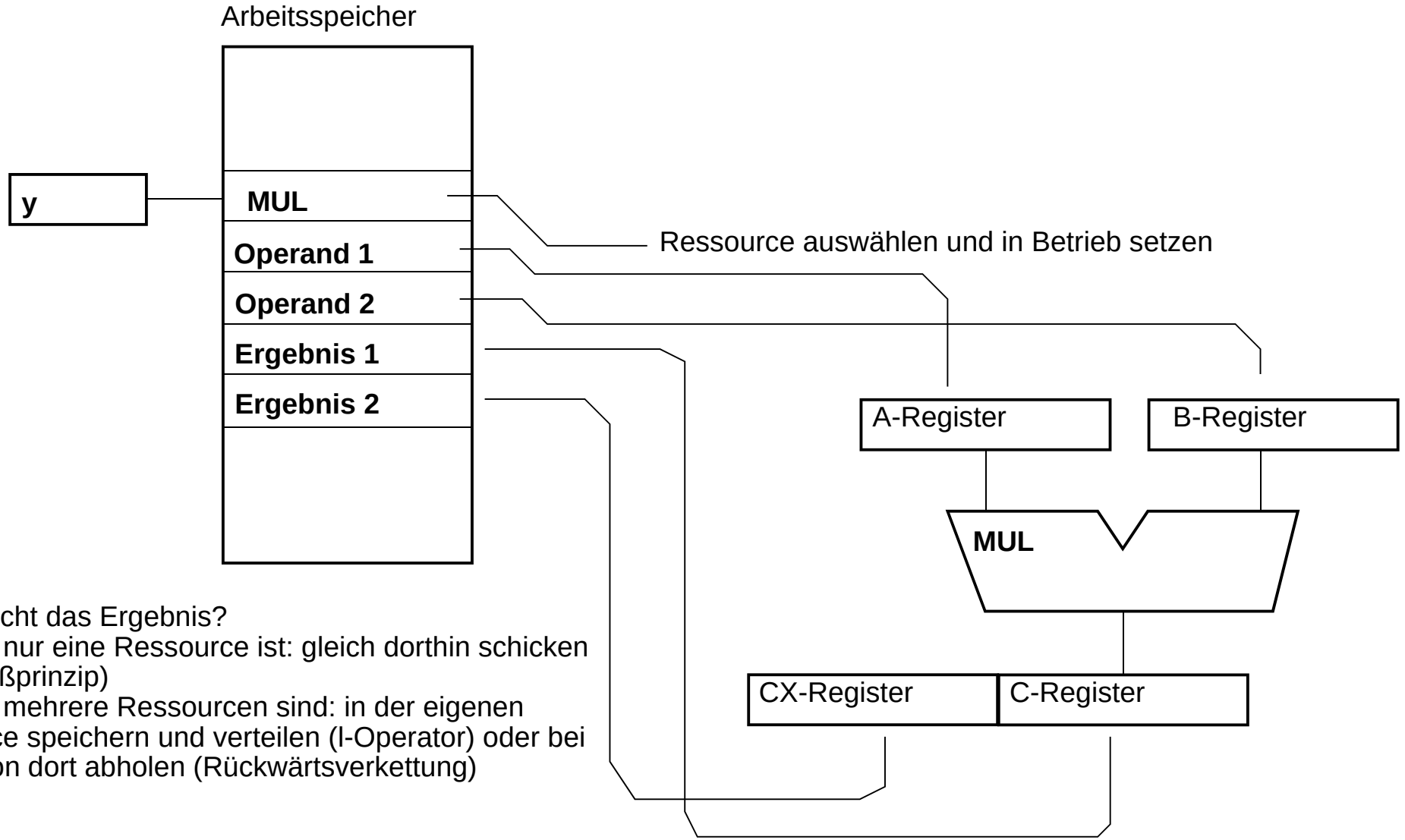
Hierzu korrespondiert jeweils ein
Block von Wörtern im Speicher.
Sie fallen 1:1 in die Register.
Die Befehle haben nur eine Adresse
(y-Operatoren).



Registermodell Emulator

Ziel: eine überschaubare,
herkömmliche Maschine, die die
ReAl-Prinzipien emulieren kann.

7. 3. 2014



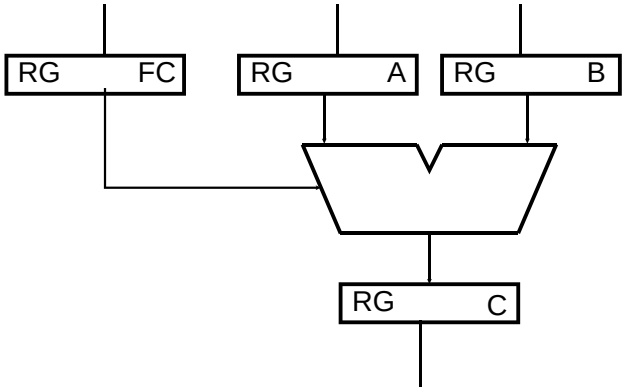
Wer braucht das Ergebnis?
Wenn es nur eine Ressource ist: gleich dorthin schicken (Datenflußprinzip)
Wenn es mehrere Ressourcen sind: in der eigenen Ressource speichern und verteilen (I-Operator) oder bei Bedarf von dort abholen (Rückwärtsverkettung)

Wo gehen die Ergebnisse hin?
Zu den Eingängen zurück (Akkumulatorprinzip)
In eigene Speicher (Ausgangsregister)
In die Ressourcen, die damit weiterarbeiten (Verkettung).

ReAl
Befehlsausführung
So wirkt der y-Operator

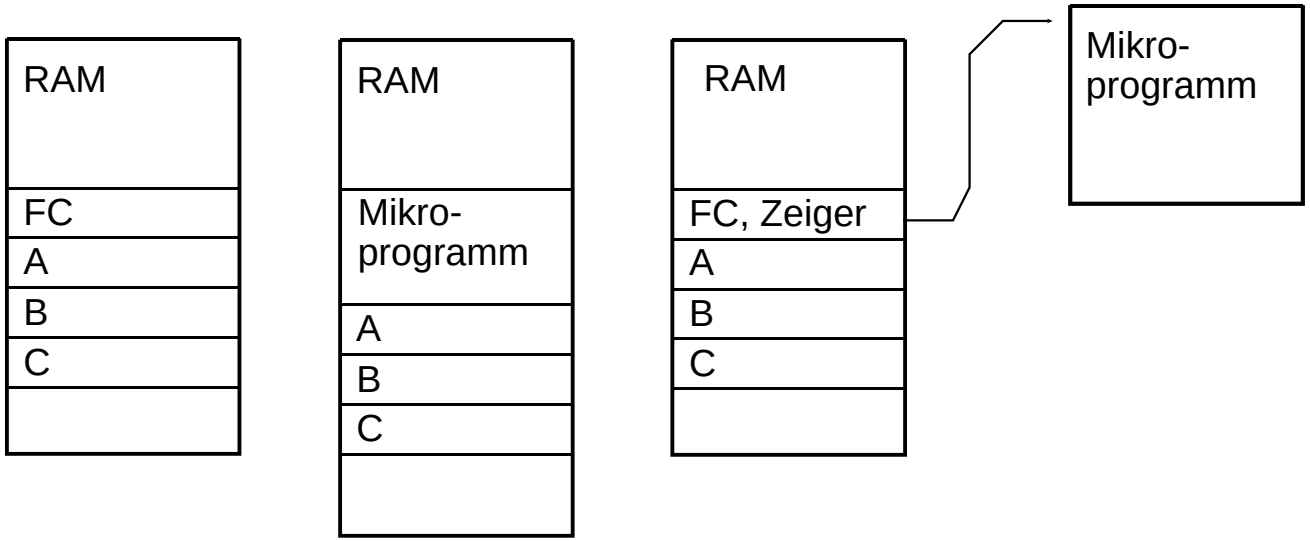
7. 3. 2014

a) Die Ressource ist eine Schaltung



Der Y-Operator setzt die Schaltung in Betrieb (Startsignal)

b) Die Ressource ist ein Speicherbereich mit Parametern

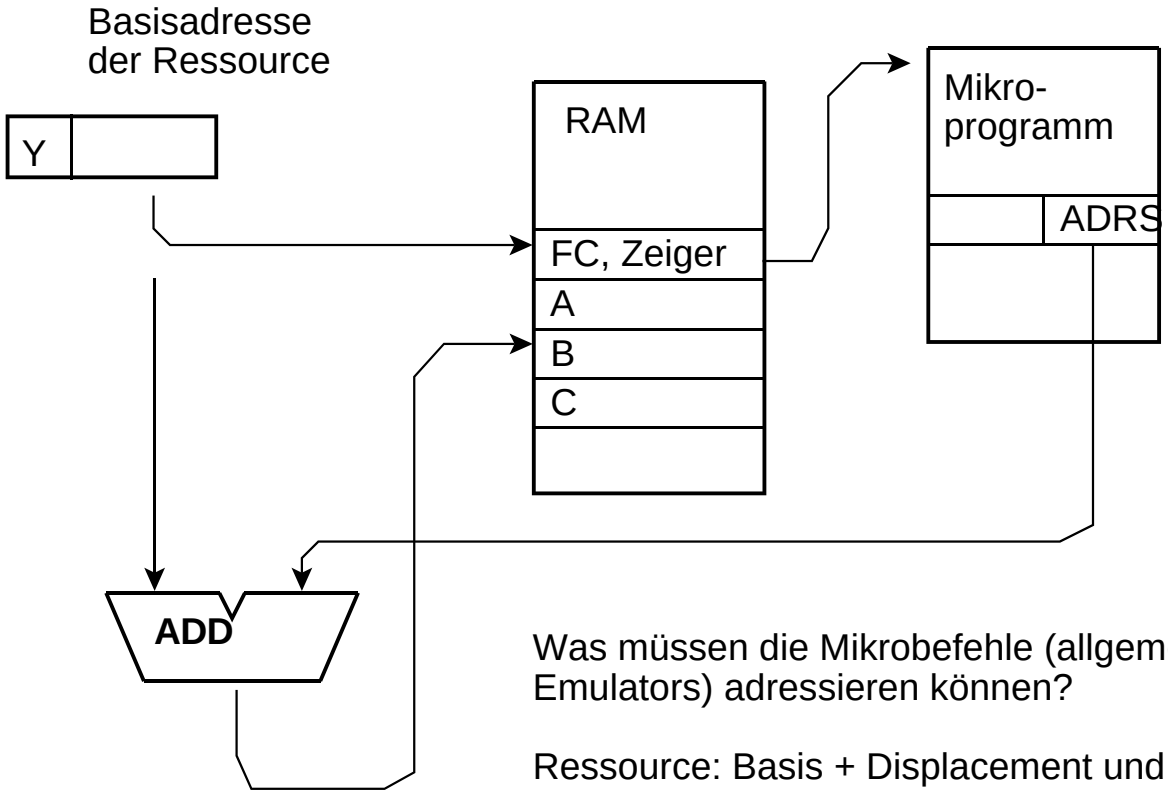


Der Y-Operator adressiert einen Funktionscode. Der setzt eine Folgesteuerung in Gang (Steuerkette).

Es ist kein einfacher Funktionscode, sondern ein Mikroprogramm.

Das Mikroprogramm ist nur ein einziges Mal vorhanden. Dann enthält der Funktionscode einen Zeiger.

7. 3. 2014

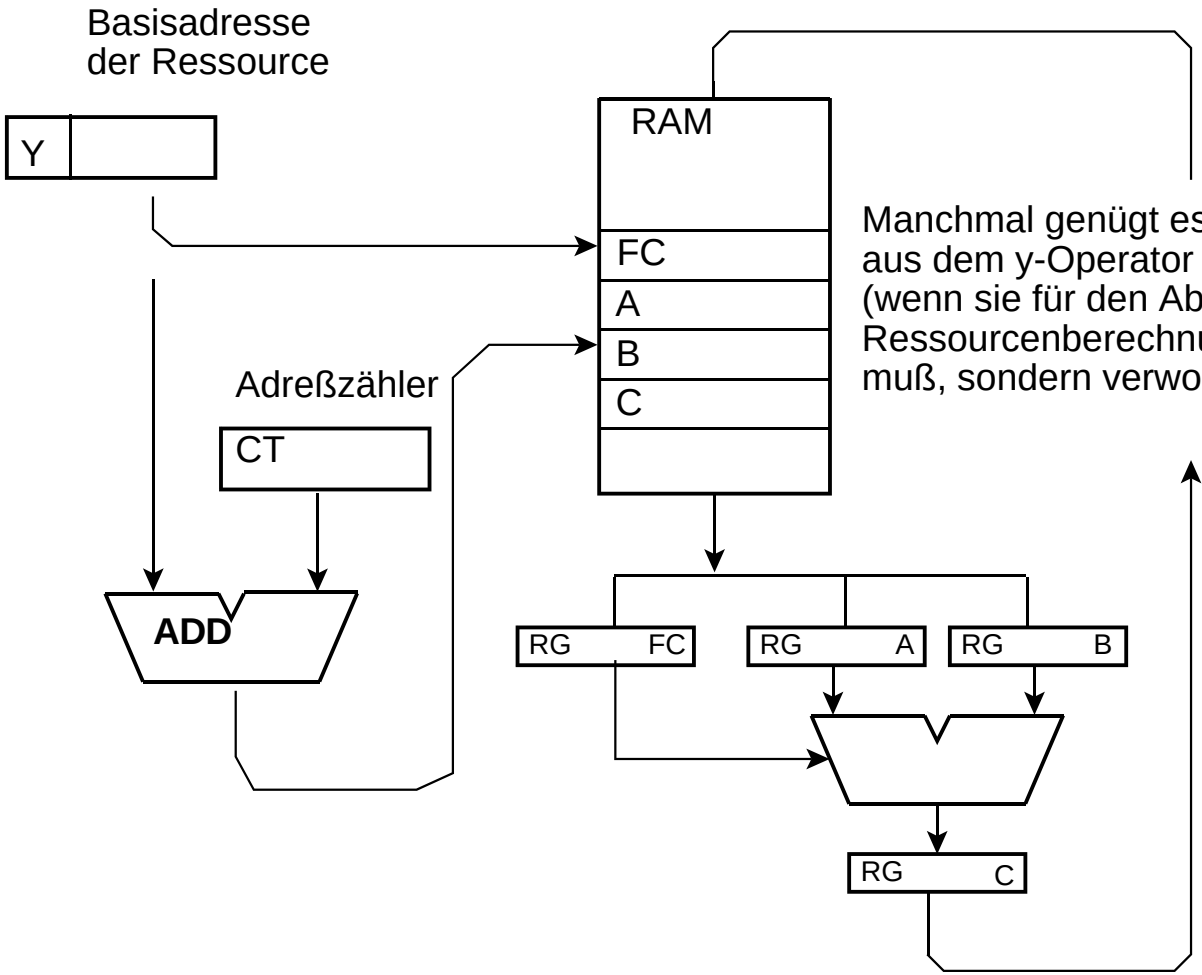


Was müssen die Mikrobefehle (allgemein: die Befehle des Emulators) adressieren können?

Ressource: Basis + Displacement und die Hardwareregister.

Basis + Displacement Ressource, Basis + Displacement eines Arbeitsbereichs.

7. 3. 2014



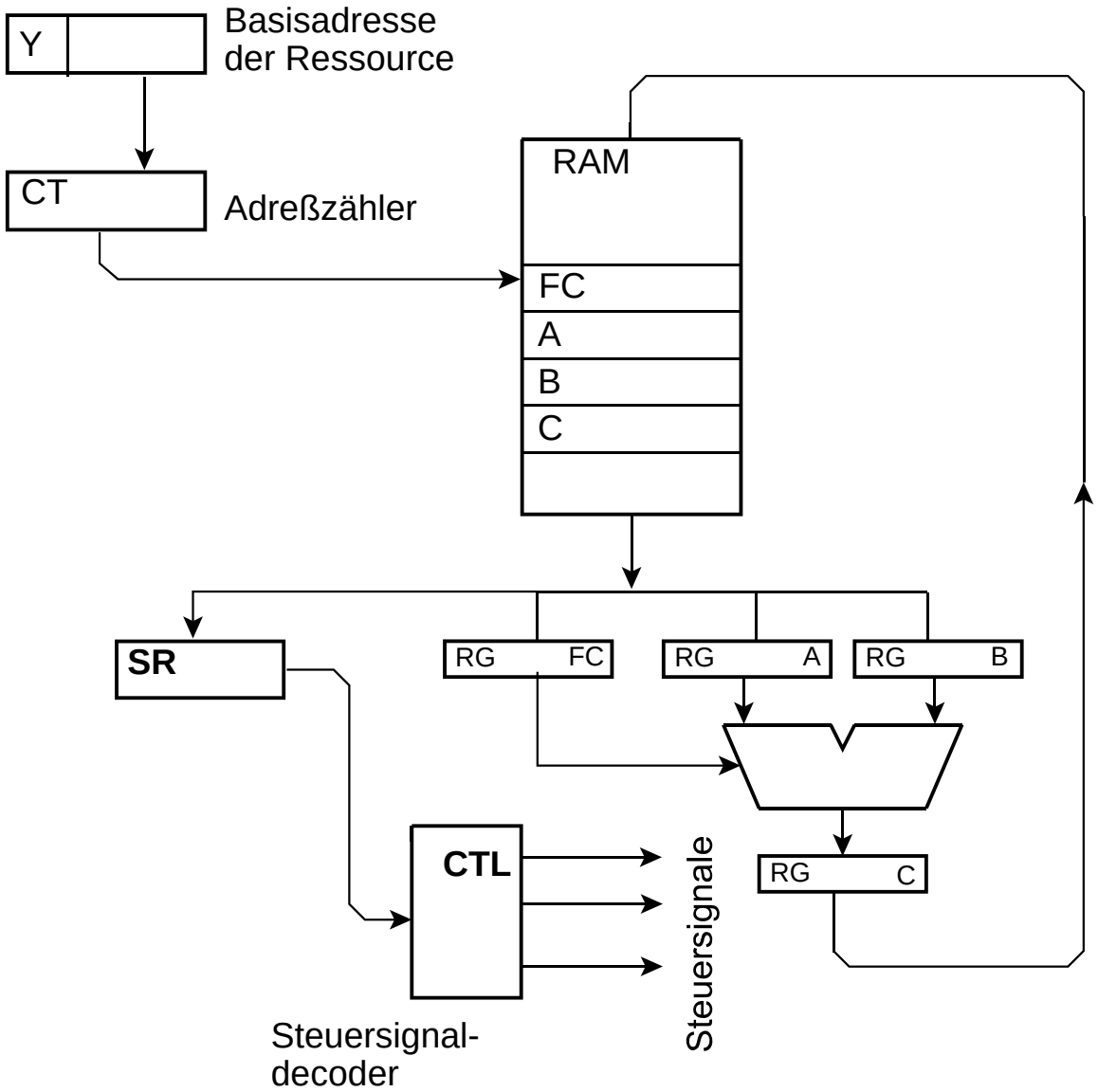
Manchmal genügt es auch, die Basisadresse aus dem y-Operator einfach hochzuzählen (wenn sie für den Ablauf der Ressourcenberechnung nicht erhalten bleiben muß, sondern verworfen werden kann).

Varianten:

a) Im FC steht eine Art Mikrobefehl, der die Folge der zu ladenden und abzuspeichernden Hardwareregister angibt.

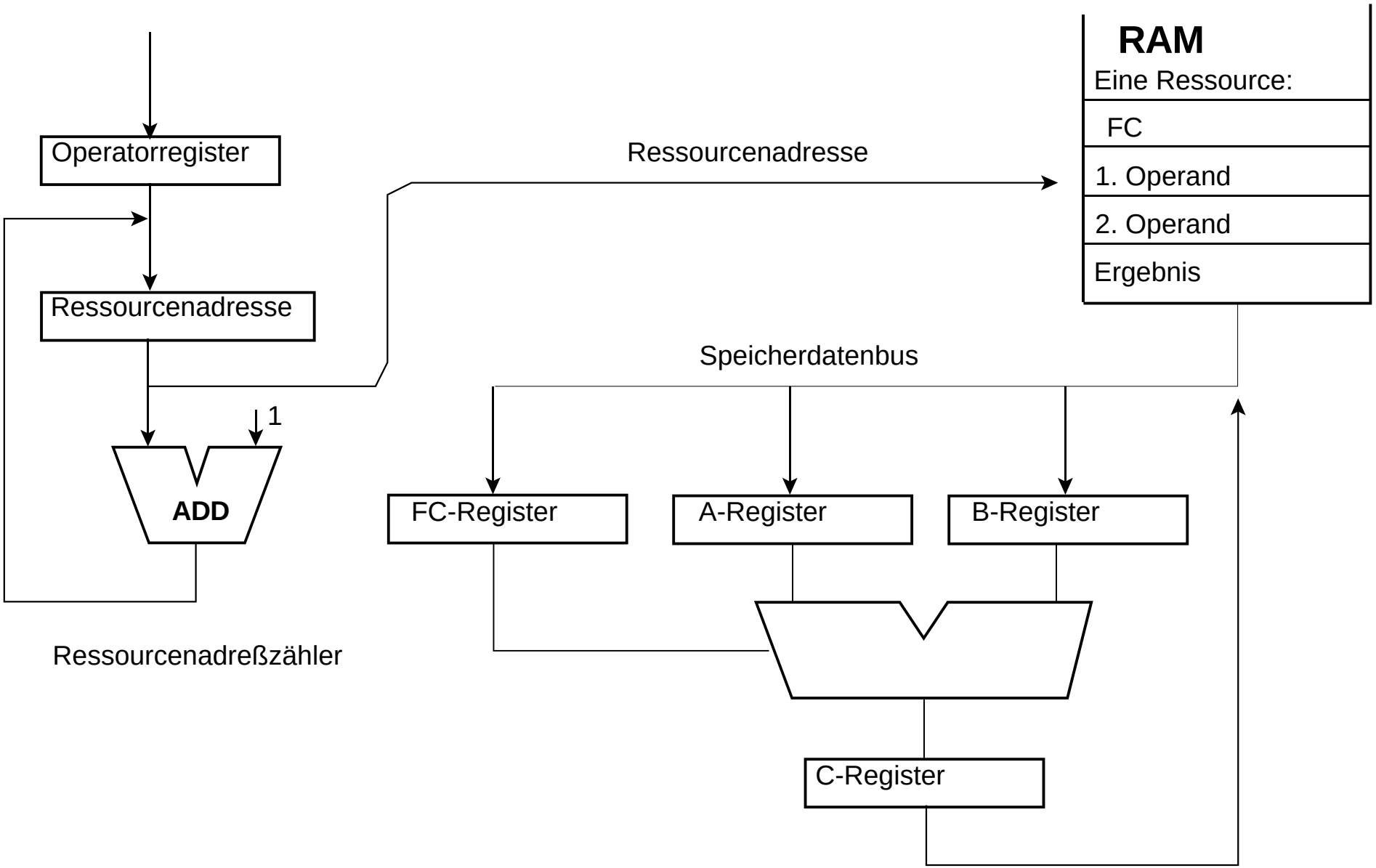
b) Für jeden Ressourcentyp gibt es eine Steuerkette in der Hardware. Der FC wählt die jeweilge Steuerkette aus.

Der einfachste Fall: Die Daten fallen aus dem Speicher direkt in die Hardware. Wenn es 1:1-Transporte sind (Implementierungseffizienz = 1), genügt eine einfache Folgesteuerung mit Adreßzähler.



Der Mikrobefehl im FC besteht aus Steueranweisungen, die parallel in ein Schieberegister geladen und Takt für Takt ausgeschoben werden.

7. 3. 2014



ReAl Emulator

7. 3. 2014