

Aufgabenblatt 4: Von Neumann Architektur und Mikroprogrammierwerk

Aufgabe 1: Von Neumann Interpretationszyklus

- a) In Abbildung 1) sehen Sie die in der Vorlesung vorgestellte von Neumann Architektur. Es sollen nun unter Berücksichtigung des aktuellen Zustandes (PC, Hauptspeicher, Register, etc.) die beiden nächsten Interpretationszyklen schrittweise durchlaufen werden. Dabei handelt es sich um einen Increment- und um einen Store-Befehl.

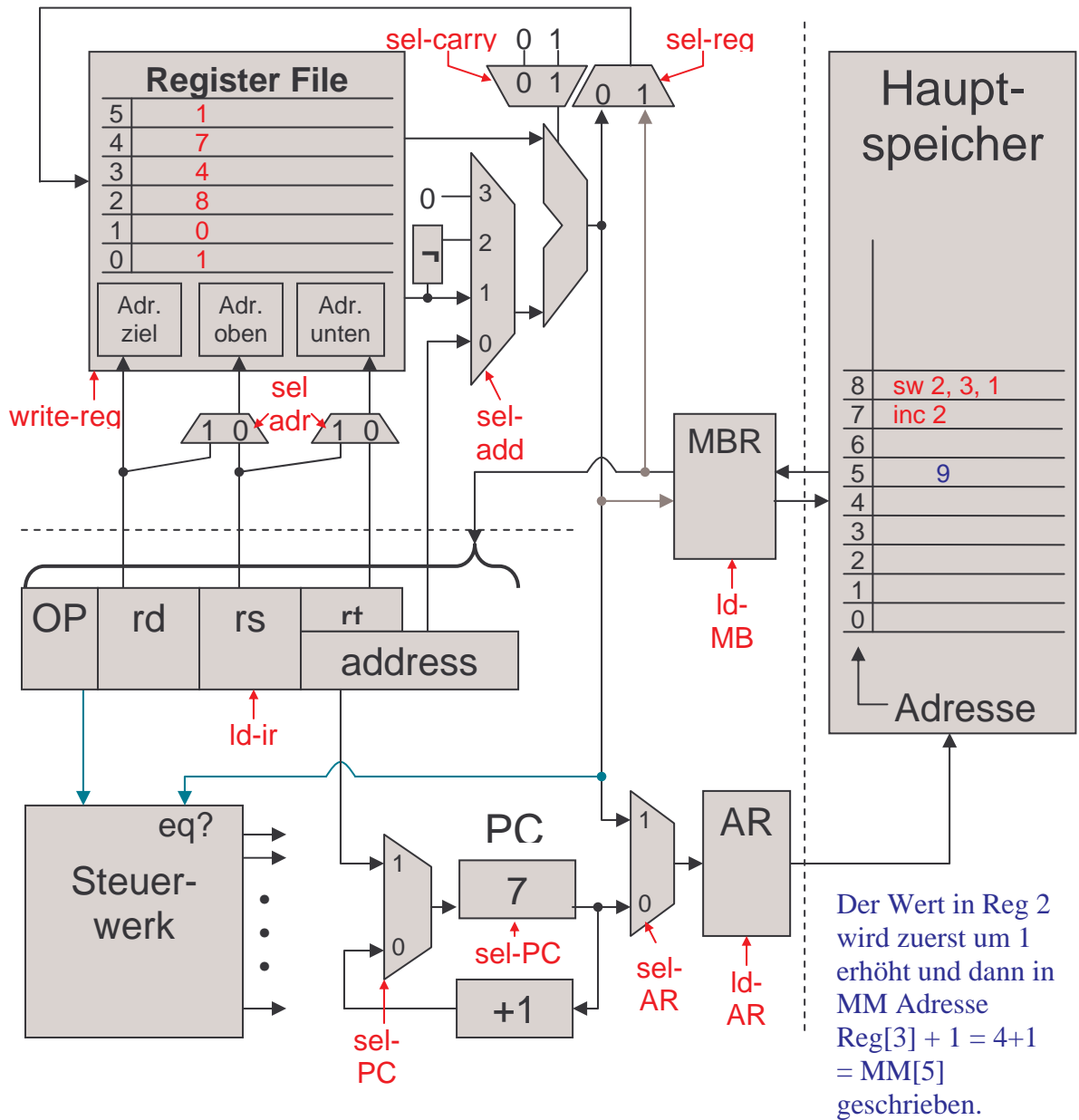


Abbildung 1) Von Neumann Architektur

Schritte:
AR := PC
MBR := MM[AR]
IR := MBR
PC := PC + 1
R[rd] := R[rs]+R[rt]
AR := PC
MBR := MM[AR]
IR := MBR
PC := PC + 1
AR := R[rs] + addr
MBR := R[rd]
MM[AR] := MBR

Instruktion Inc 2:

Erhöhe den Inhalt des Registerfiles an der Adresse 2 um den Wert 1.

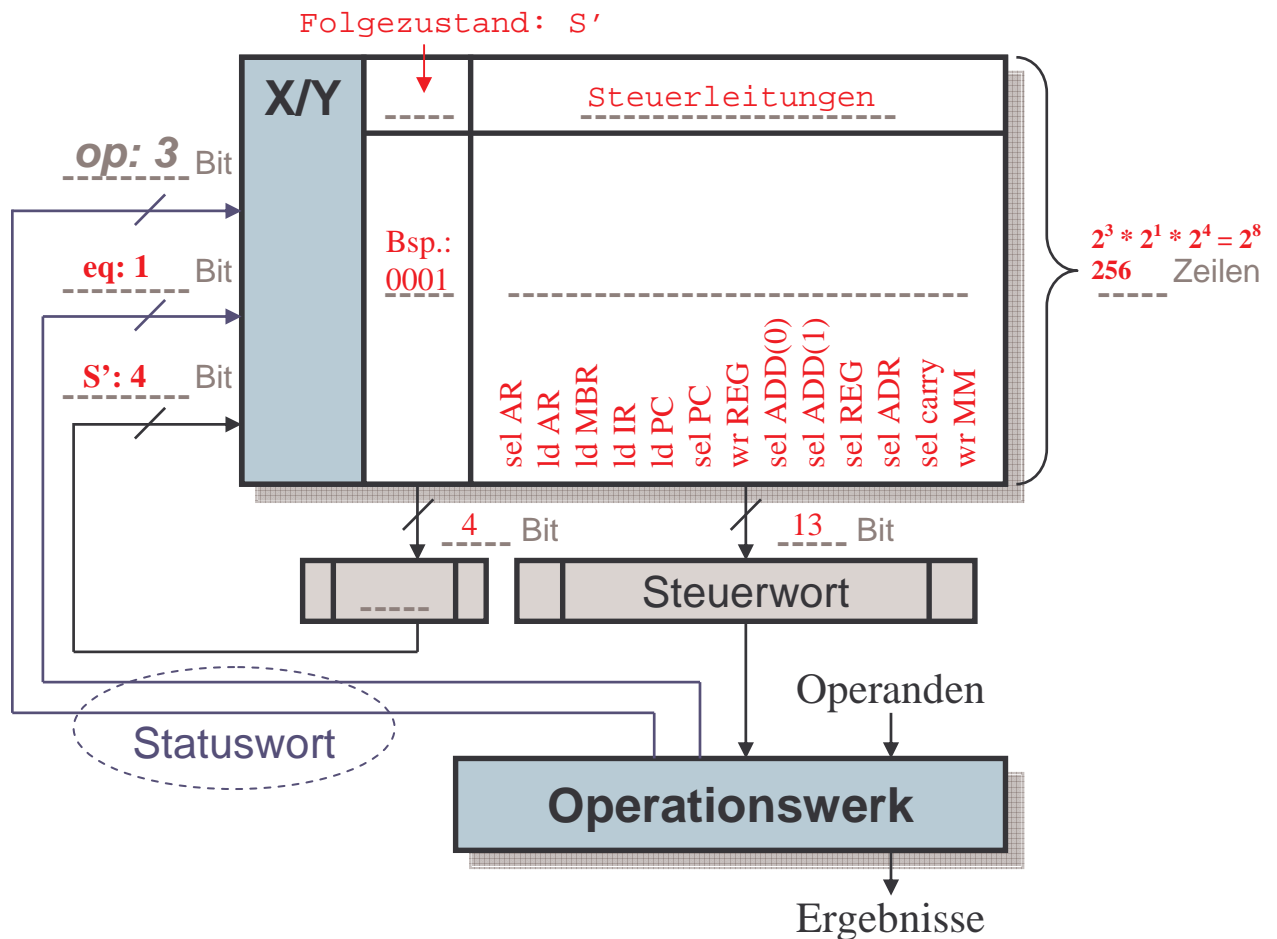
Instruktion sw 2, 3, 1:

Speichere den Inhalt des Registerfiles mit der Adresse rd in den Speicher an der Adresse R[rs] + addr. Dabei ist R[rs] der Inhalt des Registerfiles mit der Adresse rs.

Aufgabe 2: Mikroprogrammierwerk (realisiert im Mikroprogramm Speicher)

In Abbildung 2) soll der aus der Vorlesung bekannte Automat des Von Neumann'schen Interpretationszyklus als Mikroprogramm umgesetzt werden. Dabei müssen Steuersignale für das Operationswerk erzeugt werden. Dazu werden Statussignale des Operationswerkes als Eingabe des Mikroprogramm Speichers abgefragt.

- a) Füllen Sie die gestrichelten Linien in Abbildung 2) mit den Werten aus, die sich aus einer Realisierung als Mikroprogramm im Mikroprogramm Speicher ergeben würden.



REDUNDANZ: *Op* wird als Eingabe nur in einem und *eq* nur in 2 Zuständen abgefragt. Als **Teil der Adressbildung** sind aber in allen Zuständen *S* auch immer alle möglichen Werte von *op* und *eq* abzudecken. So existieren $2^3 + 2^1 = 16$ Zeilen für jeden Zustand im Mikroprogramm Speicher. Im Zustand $S = 0000$ sind diese 16 Zeilen alle identisch, da weder *op* noch *eq* abgefragt werden und daher für all ihre Wertkombinationen das gleiche gemacht werden muss.

- b) - Beurteilen Sie Ihr Ergebnis unter den Gesichtspunkten:

- Platzverbrauch (Größe des Mikroprogramm Speichers) Groß: 17 Bit * 256 Zeilen
= 17 * 32 Byte = 544 Byte
- Performance Schnellste Lösung, die als Mikroprogrammierwerk möglich ist.

- Handelt es sich hierbei um einen sinnvollen Kompromiss?

Wenn immer alle Eingaben benötigt werden (vgl. Und-Funktion), ist das eine optimale Lösung ohne Redundanz.

- Wie könnte eine mögliche Verbesserung aussehen?

Ist sinnvoll, wenn der Speicherplatz keine Rolle spielt, die Lösung schnell und einfach im Aufbau sein soll.
→ Auslagern der Redundanten Eingaben in separate Speicher oder sogar eigene Mikroprogrammierwerke.

Aufgabe 3: Mikroprogrammierwerk (mit Schaltnetz für Adressberechnung)

In Abbildung 3) wurde das in Aufgabe 2 vorgestellte Mikroprogrammierwerk um ein Schaltnetz für die Adressberechnung erweitert.

- a) Füllen Sie die gestrichelten Linien und die Registerfiles in Abbildung 3) mit den Werten aus, die sich aus einer Realisierung als Mikroprogramm mit einem Schaltnetz für die Adressberechnung ergeben würden.

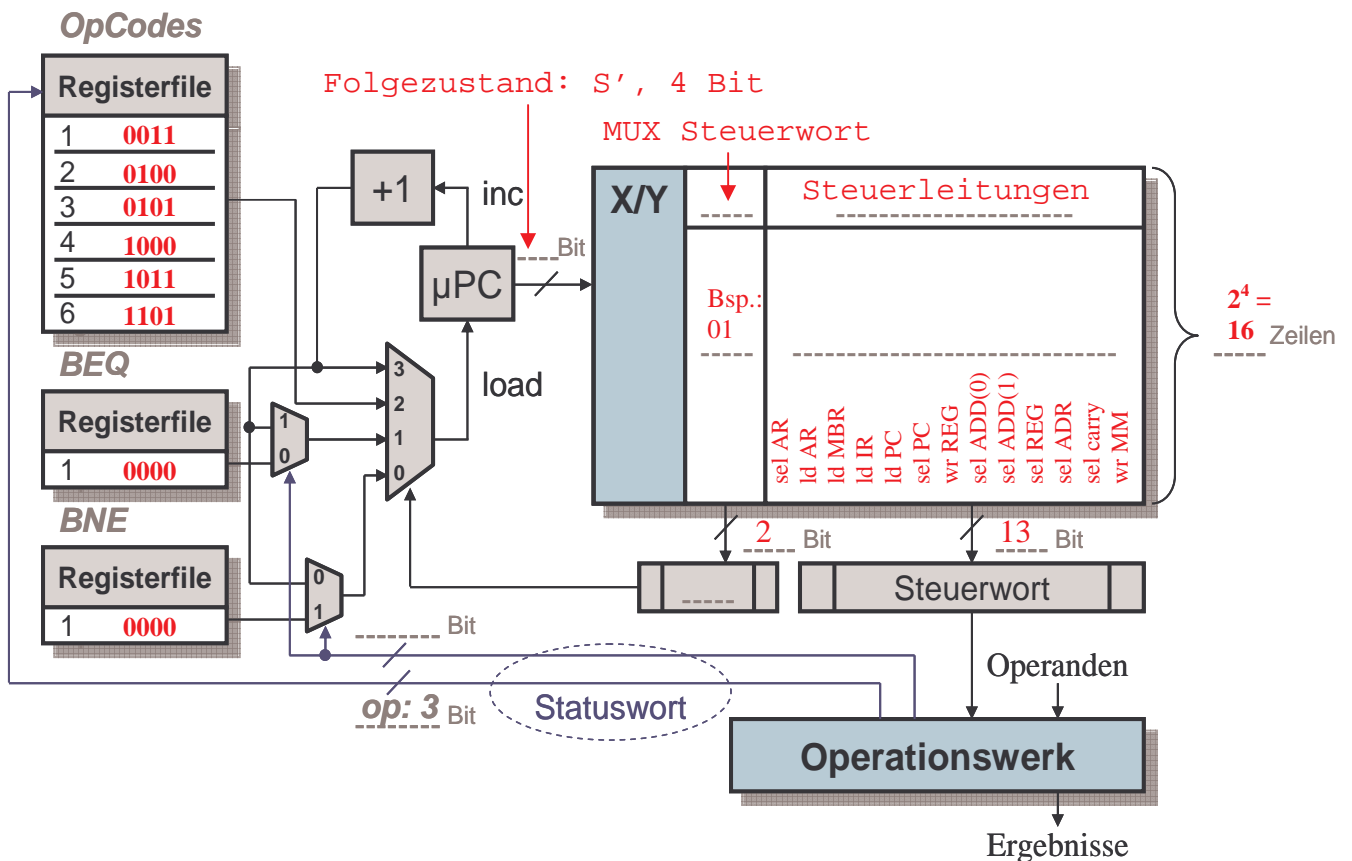


Abbildung 3) Mikroprogrammierwerk

- b) - Beurteilen Sie Ihr Ergebnis unter den Gesichtspunkten:

- Platzverbrauch (Gesamter Bedarf an Speicher und Schaltlogik)
 - Klein: $15 \text{ Bit} * 16 \text{ Zeilen} + (6 + 1 + 1) \text{ Register} = 31 \text{ Byte}$
 - + 3 Multiplexer + ProgrammCounter + Incrementierer
- Performance

Langsamer als in Aufgabe 2), da die Adressberechnung immer zunächst ausgeführt werden muss. Die endgültige Berechnung kann je nach Operation (op, eq) erst nach der Aktion im Operationswerk ausgeführt werden.

- Vergleichen Sie das Ergebnis mit dem Resultat aus Aufgabe 2.

Für komplexe Aufgaben, wie z.B. Die komplette Realisierung eines Steuerwerkes wird die hier in Aufgabe 3) vorgestellte Variante bevorzugt, da Sie keine Redundanz beinhaltet. Einfache Operationen, wie z.B. eine Und-Funktion, bei denen sowieso immer der gesamte Input ausgewertet werden muss, lassen sich einfacher in der Variante aus Aufgabe 2) realisieren.