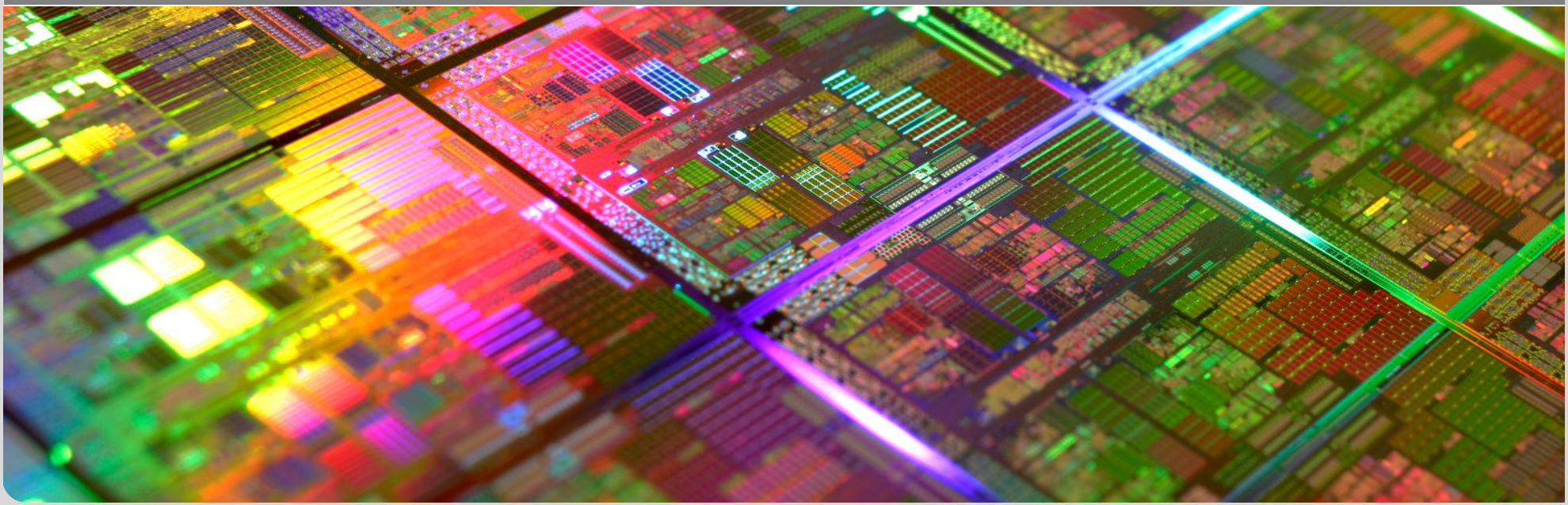
Karlsruher Institut für Technologie

RO-Tutorien 17 und 18

Tutorien zur Vorlesung “Rechnerorganisation”

Christian A. Mandery

TUTORIENWOCHE 4 AM 24.05.2012



Heute

- Komponenten eines Von-Neumann-Rechners
- Einführung in MIMA
- Übungsaufgaben

Die Von-Neumann-Architektur

- Von-Neumann-Architektur bezieht sich auf Rechner, wir beschäftigen uns hier aber zuerst einmal nur mit der CPU
- Besteht aus verschiedenen Komponenten:
  - Steuerwerk
  - Rechenwerk
  - Speicherwerk
  - Registersatz
  - Systembusschnittstelle
- Programme und Daten liegen im selben Speicher
  - Müssen über den gleichen Speicherbus angesprochen werden (Von-Neumann-Flaschenhals)
  - Alternative: Harvard-Architektur mit getrennten Speicher

Die Von-Neumann-Architektur

- Von-Neumann-Architektur bezieht sich auf Rechner, wir beschäftigen uns hier aber zuerst einmal nur mit der CPU
- Besteht aus verschiedenen Komponenten:
  - Steuerwerk
  - Rechenwerk
  - Speicherwerk
  - Registersatz
  - Systembusschnittstelle
- Programme und Daten liegen im selben Speicher
  - Müssen über den gleichen Speicherbus angesprochen werden (Von-Neumann-Flaschenhals)
  - Alternative: Harvard-Architektur mit getrennten Speicher

Steuerwerk

- Für die Abarbeitung des Programms zuständig
  1. Läd den auszuführenden Befehl in das Befehlsregister (**Holphase**)
  2. Dekodiert den Befehl im Befehlsregister (**Dekodierphase**)
  3. Führt den dekodierten Befehl unter Verwendung der anderen CPU-Komponenten aus (**Ausführungsphase**)
- Steuerregister beeinflusst Verhalten des Steuerwerks (Beispiele?)
- Fest verdrahtete Logik vs. Mikroprogrammierung (Vor-/Nachteile?)

Steuerwerk

- Für die Abarbeitung des Programms zuständig
  1. Läd den auszuführenden Befehl in das Befehlsregister (**Holphase**)
  2. Dekodiert den Befehl im Befehlsregister (**Dekodierphase**)
  3. Führt den dekodierten Befehl unter Verwendung der anderen CPU-Komponenten aus (**Ausführungsphase**)
- Steuerregister beeinflusst Verhalten des Steuerwerks (Beispiele?)
- Fest verdrahtete Logik vs. Mikroprogrammierung (Vor-/Nachteile?)

Steuerwerk

- Für die Abarbeitung des Programms zuständig
  1. Läd den auszuführenden Befehl in das Befehlsregister (**Holphase**)
  2. Dekodiert den Befehl im Befehlsregister (**Dekodierphase**)
  3. Führt den dekodierten Befehl unter Verwendung der anderen CPU-Komponenten aus (**Ausführungsphase**)
- Steuerregister beeinflusst Verhalten des Steuerwerks (Beispiele?)
- Fest verdrahtete Logik vs. Mikroprogrammierung (Vor-/Nachteile?)

Rechenwerk

- ALU = “arithmetic logic unit”
- Führt auf Anweisung des Steuerwerks arithmetisch-logische Operationen aus
- Statusregister speichert zusätzliche Informationen über die letzte Berechnung, z.B.
  - Negative Flag
  - Carry Flag
  - Overflow Flag

Speicherwerk (Adresswerk)

- Beinhaltet in Von-Neumann-Architektur *eigentlich* auch den Speicher selbst, wir betrachten erstmal nur die CPU
- Regelt den Zugriff auf den Hauptspeicher, der Programme und Daten enthält
- Bei vielen modernen Prozessoren zusätzliche Aufgabenbereiche für MMU (“memory management unit”) des Prozessors, z.B.
  - Zugriffsschutz
  - Segmentierung
  - Paging (Kachelverwaltung)

Registersatz

- Enthält die Register des Prozessors
- Spezialregister vs. allgemein verwendbare (general purpose) Register
- Was ist ein Register?
- Welche Register kennen wir?

Registersatz

- Enthält die Register des Prozessors
- Spezialregister vs. allgemein verwendbare (general purpose) Register
- Was ist ein Register?
- Welche Register kennen wir?

Registersatz

- Enthält die Register des Prozessors
- Spezialregister vs. allgemein verwendbare (general purpose) Register
- Was ist ein Register?
- Welche Register kennen wir?

Systembusschnittstelle

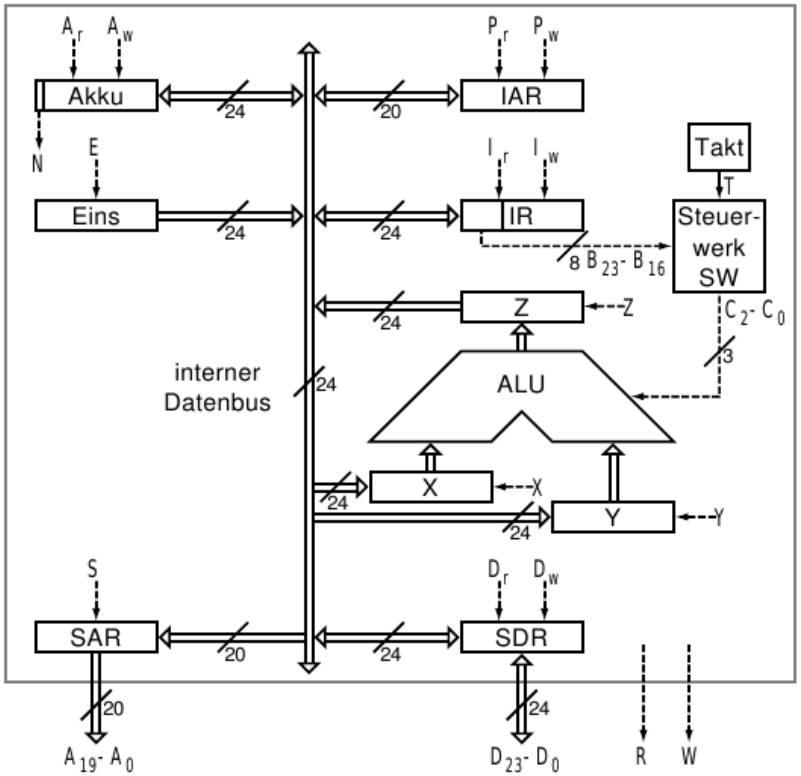
- Erlaubt dem Prozessor, über den Systembus mit anderen Rechnerkomponenten zu kommunizieren
- Bestehend aus:
  - Steuerbus (unidirektional): Wahl des Betriebsmodus
  - Adressbus (unidirektional): Wahl der Adresse
  - Datenbus (bidirektional): Übergabe des Datenworts
- Beispiele:
  - “Schreibe den Wert 10 bei Adresse 20” (CPU $\rightarrow$ RAM)
  - “Lies den Wert bei Adresse 20” (CPU $\rightarrow$ RAM) - “10” (RAM $\rightarrow$ CPU)

Einführung in MIMA

- Mikroprogrammierte Minimalmaschine
- Sehr einfaches Modell für eine CPU mit Von-Neumann-Architektur
- Wurde im Rahmen der Rechnerorganisation-Vorlesung entwickelt und wird dort vorgestellt
- Folgende Spezifikationen gibt es als “Merkblatt” auch nochmal an das 4. Übungsblatt angehängt und auf der Vorlesungshomepage zum Download (als mima.pdf)

Die Architektur der MIMA

Architektur der MIMA



Register

Akku: Akkumulator
X: 1. ALU Operand
Y: 2. ALU Operand
Z: ALU Ergebnis
Eins: Konstante 1
IAR: Instruktionsadreßregister
IR: Instruktionsregister
SAR: Speicheradreßregister
SDR: Speicherdatenregister

Steuersignale vom SW

– für den internen Datenbus

A_r: Akku liest
A_w: Akku schreibt
X: X-Register liest
Y: Y-Register liest
Z: Z-Register schreibt
E: Eins-Register schreibt
P_r: IAR liest
P_w: IAR schreibt
I_r: IR liest
I_w: IR schreibt
D_r: SDR liest
D_w: SDR schreibt
S: SAR liest

– für die ALU

C_2-C_0: Operation auswählen

– für den Speicher

R: Leseanforderung
W: Schreibanforderung

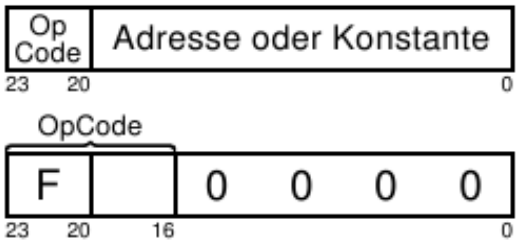
Meldesignale zum SW

T: Takteingang
N: Vorzeichen des Akku
B_{23}-B_{16}: OpCode-Feld im IR

Befehle der MIMA

OpCode	*Mnemonik*	*Beschreibung*
0	LDC c	c -> Akku
1	LDV a	<a> -> Akku
2	STV a	Akku -> <a>
3	ADD a	Akku + <a> -> Akku
4	AND a	Akku AND <a> -> Akku
5	OR a	Akku OR <a> -> Akku
6	XOR a	Akku XOR <a> -> Akku
7	EQL a	falls Akku = <a>:-1 -> Akku
		sonst : 0 -> Akku
8	JMP a	a -> IAR
9	JMN a	falls Akku < 0 : a -> IAR
F0	HALT	stoppt die MIMA
F1	NOT	bilde Eins-Komplement von Akku -> Akku
F2	RAR	rotiere Akku eins nach rechts -> Akku

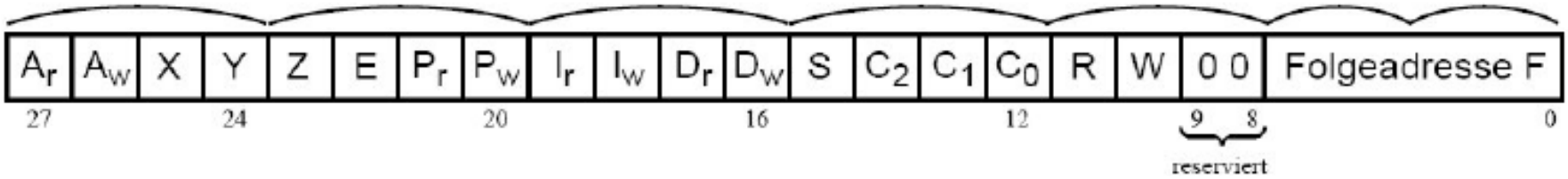
Befehlsformate



ALU-Operationen der MIMA

$C_2 C_1 C_0$	*ALU Operation*
0 0 0	tue nichts (d.h. Z -> Z)
0 0 1	X + Y -> Z
0 1 0	rotiere X nach rechts -> Z
0 1 1	X AND Y -> Z
1 0 0	X OR Y -> Z
1 0 1	X XOR Y -> Z
1 1 0	Eins-Komplement von X -> Z
1 1 1	falls X = Y, -1 -> Z, sonst 0 -> Z

Mikrobefehlsformat der MIMA



Mikroprogrammierung

- Das MIMA-Steuerwerk ist mikroprogrammiert
- $\Rightarrow$ Ein Maschinenbefehl wird durch ein kleines Programm von Mikrobefehlen ausgeführt
- Jeder dieser Mikrobefehle gibt an, welche der Steuerleitungen in einem Taktzyklus aktiv sind
- Register-Transfer-Anweisungen als alternative Schreibweise:
  - $A_w = 1, P_r = 1$ (Steuersignale)
  - $Akku \rightarrow IAR$ (Register-Transfer)

Mikroprogrammierung

- Das MIMA-Steuerwerk ist mikroprogrammiert
- $\Rightarrow$ Ein Maschinenbefehl wird durch ein kleines Programm von Mikrobefehlen ausgeführt
- Jeder dieser Mikrobefehle gibt an, welche der Steuerleitungen in einem Taktzyklus aktiv sind
- Register-Transfer-Anweisungen als alternative Schreibweise:
  - $A_w = 1, P_r = 1$ (Steuersignale)
  - $Akku \rightarrow IAR$ (Register-Transfer)

Übungsaufgabe 1: Mikrocode #1

Geben Sie das Mikroprogramm für die Holphase und Ausführungsphase des MIMA-Befehls

`OR a` `# akku OR <a>` $\rightarrow$ `Akku`

1. in Register-Transfer-Anweisung Schreibweise
2. in binärer Schreibweise (nur Takte 1 und 7)

an.

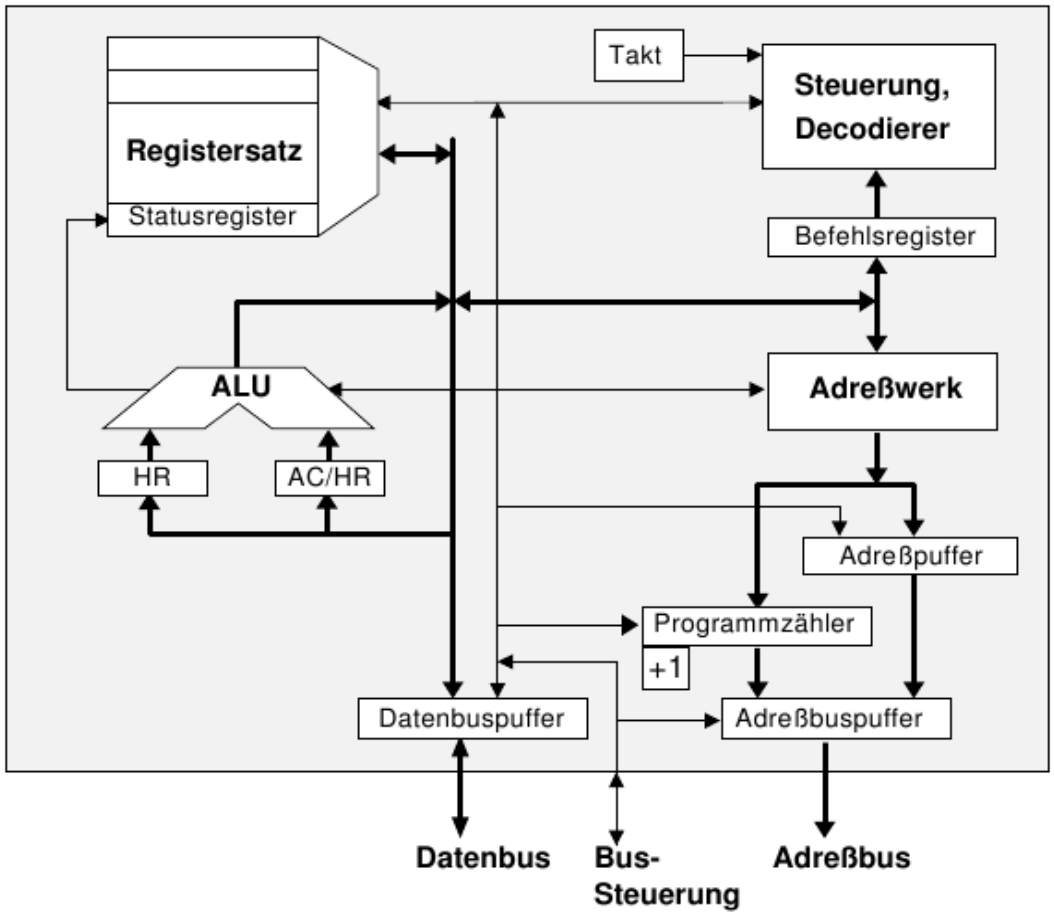
Übungsaufgabe 2: Befehlsdekodierung

Das bisher reservierte 8. Bit im Mikrobefehlsformat der MIMA sei mit D bezeichnet und wird als Kennzeichen dafür verwendet, dass die Adresse des nächsten Mikrobefehls aus dem Befehlsteil $B_{23} - B_{16}$ ermittelt werden muss.
Wie sieht dann der Mikrobefehl für die Dekodierung (6. Takt) aus?

Übungsaufgabe 3: Mikrocode #2

Schreiben Sie ein Programm in Register-Transfer-Schreibweise, das den Inhalt des Akkumulators als Zweierkomplement wieder im Akkumulator speichert. Beginnen Sie dabei ab Takt 7 der Befehlsabarbeitung.

Übungsaufgabe 4: Bus-Architektur



Übungsaufgabe 4: Aufgaben

1. Nehmen Sie an, dass das Befehlsregister als Warteschlange (OpCode prefetch queue) realisiert ist.
   Ändern Sie die Bus-Architektur so, dass parallel zu Aktionen im Rechenwerk und Registersatz ein OpCode Prefetching möglich ist.
2. Während das Ergebnis der zuletzt ausgeführten Operation im Rechenwerk in den Registersatz transportiert wird, soll ein Hilfsregister der ALU mit dem nächsten Operanden aus dem Datenbuspuffer oder aus dem Registersatz (sofern nicht das gleiche Register hierfür benötigt wird) geladen werden. Ändern Sie die Bus-Architektur so, dass diese Parallelisierung im Rechenwerk möglich wird.
3. Ändern Sie die Bus-Architektur so, dass zwei Operanden parallel an die Eingänge der ALU geführt werden können. Dabei soll der eine Operand immer aus dem Registersatz stammen, während der andere Operand wahlweise auf dem Registersatz oder dem Datenbuspuffer stammt.

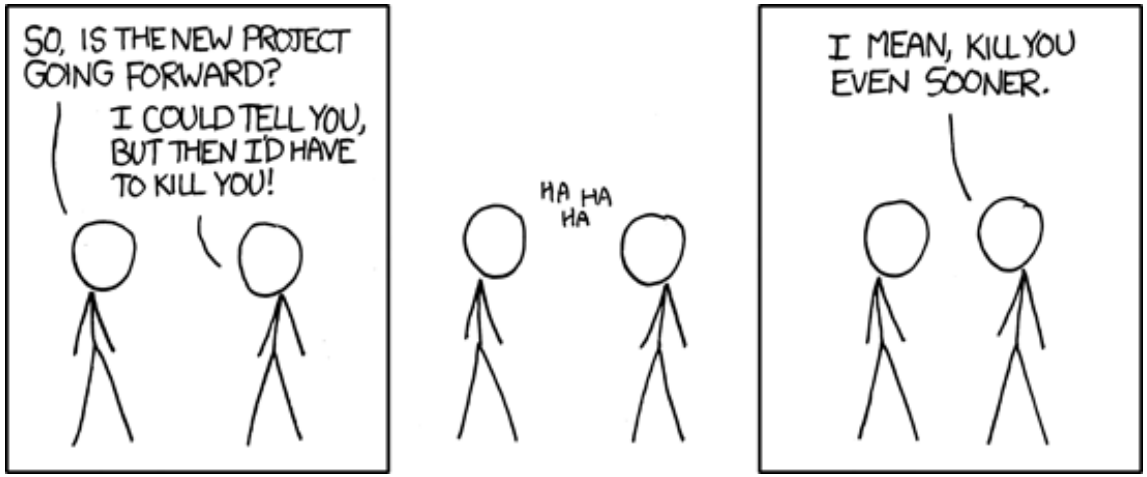
Übungsaufgabe 4: Aufgaben

1. Nehmen Sie an, dass das Befehlsregister als Warteschlange (OpCode prefetch queue) realisiert ist.
   Ändern Sie die Bus-Architektur so, dass parallel zu Aktionen im Rechenwerk und Registersatz ein OpCode Prefetching möglich ist.
2. Während das Ergebnis der zuletzt ausgeführten Operation im Rechenwerk in den Registersatz transportiert wird, soll ein Hilfsregister der ALU mit dem nächsten Operanden aus dem Datenbuspuffer oder aus dem Registersatz (sofern nicht das gleiche Register hierfür benötigt wird) geladen werden.
   Ändern Sie die Bus-Architektur so, dass diese Parallelisierung im Rechenwerk möglich wird.
3. Ändern Sie die Bus-Architektur so, dass zwei Operanden parallel an die Eingänge der ALU geführt werden können. Dabei soll der eine Operand immer aus dem Registersatz stammen, während der andere Operand wahlweise auf dem Registersatz oder dem Datenbuspuffer stammt.

Übungsaufgabe 4: Aufgaben

1. Nehmen Sie an, dass das Befehlsregister als Warteschlange (OpCode prefetch queue) realisiert ist.
   Ändern Sie die Bus-Architektur so, dass parallel zu Aktionen im Rechenwerk und Registersatz ein OpCode Prefetching möglich ist.
2. Während das Ergebnis der zuletzt ausgeführten Operation im Rechenwerk in den Registersatz transportiert wird, soll ein Hilfsregister der ALU mit dem nächsten Operanden aus dem Datenbuspuffer oder aus dem Registersatz (sofern nicht das gleiche Register hierfür benötigt wird) geladen werden.
   Ändern Sie die Bus-Architektur so, dass diese Parallelisierung im Rechenwerk möglich wird.
3. Ändern Sie die Bus-Architektur so, dass zwei Operanden parallel an die Eingänge der ALU geführt werden können. Dabei soll der eine Operand immer aus dem Registersatz stammen, während der andere Operand wahlweise auf dem Registersatz oder dem Datenbuspuffer stammt.

Fertig!



Quelle: http://xkcd.com/707/