Hochschule für Angewandte
Wissenschaften Hof

Seminararbeit

Aufbau und Funktionsweiße eines Prozessors

Marco Vogel

unter Aufsicht von
Stefan Müller

14. November 2017

Inhaltsverzeichnis

Abbildungsverzeichnis

1 Motivation

2 Informationsverarbeitung

2.1 Binäre Darstellung von Zahlen

3 Logische Schaltglieder

3.1 AND-Gatter

3.2 OR-Gatter

3.3 NOR-Gatter

3.4 XOR-Gatter

3.5 NOT-Gatter

3.6 Flip-Flops

4 Prozessorarchitekturen

4.1 Von-Neumann Architektur

4.2 Harvard Architektur

4.3 CISC-Prozessoren

4.4 RISC-Prozessoren

5 Aufbau und Funktion

5.1 Steuerwerk

Jeder Prozessor besitzt einen gewissen Umfang ihm zur Verfügung stehender Befehle. Diese Befehle werden als Bitmuster oder Mnemonic dokumentiert. Das Steuer-

werk analysiert das Bitmuster welches aus dem Speicher zur Ausführung übergeben wird und vergleicht es mit den bekannten Bitmustern der Opcode-Befehle. Sollte eine Übereinstimmung gefunden werden wird ein Signal, welches dem dekodierten Befehl entspricht, an die angebundenen Hardware der CPU übergeben (ALU bzw. Register). Diese benutzen dieses Signal daraufhin zur weiteren Befehlsausführung.[**?**]

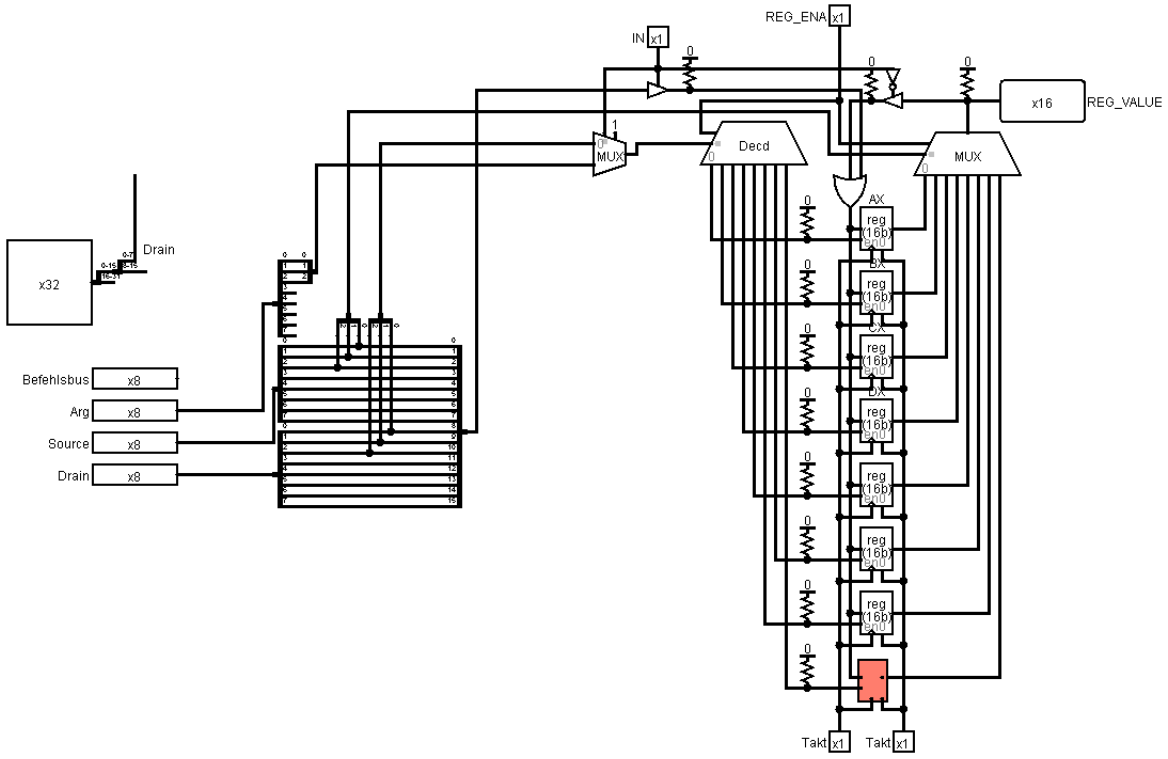

Abbildung 1: Darstellung des RegisterwerkTODO

5.2 Register

Register sind die schnellste Speichereinheit innerhalb einer CPU. Prozessoren besitzen eine vielfach höhere Ausführungsgeschwindigkeit als Arbeitsspeicher. Die CPU müsste ohne Register viele Taktzyklen auf Daten warten bevor sie diese verarbeiten könnte. Register bieten deshalb die Möglichkeit, sehr kleine Datenmengen mit einer sehr geringen Latenz prozessorintern lesen und schreiben zu können. Übliche Registergrößen sind

8,16,32 oder 64 Bit.[**?**] Sie werden aus Flip-Flops aufgebaut welche jeweils genau ein Bit speichern können, das heißt ein 64 Bit Register besteht aus 64 gemeinsam gesteuerten Flip-Flops.[**?**] Diese Art der Datenspeicherung hat allerdings auch einige Nachteile. So verbrauchen Register sehr viel Energie und Platz auf dem Prozessordie, es werden deshalb keine großen Speichermengen zur Verfügung gestellt. (Nachteile evtl streichen)

5.2.1 Universalregister

Es werden zwei Arten von Registergruppen unterschieden.In einem Universalregister kann ein Programm Werte und Variablen abspeichern.Sie stehen außerdem einem Programmierer von außen offen, das heißt er kann auf jedes Universalregister direkt zugreifen und seinen Wert verändern.

5.2.2 Spezialregister

Spezialregister werden von einer CPU für interne Zwecke genutzt. Oft sind in Prozessoren ähnliche Spezialregister zu finden.

Der StackPointer(SP) ist ein Register welcher auf die aktuelle Position des Stacks im Speicher zeigt. Wenn der Befehl zu Speicherung eines Werts auf dem Stack ausgeführt wird inkrementiert die CPU automatisch, durch die interne Verschaltung des SP, den Wert des StackPointers. Dadurch zeigt das Register immer auf die nächste freie Speicheradresse im Stack.

Der InstructionPointer(IP) enthält die Adresse des nächsten Befehls im Programmspeicher der ausgeführt werden muss. Auch er wird nach der Abarbeitung eines Befehlszyklus als letzter Schritt inkrementiert. Dieses Register bietet allerdings die Möglichkeit einen anderen Wert zu laden. Das wird zur Realisierung von Sprüngen innerhalb des Programmcodes benötigt.

Das Statusregister(SR) werden zur Ausführung von bedingten Sprunganweisungen gebraucht. Sie werden auch Flagregister genannt da die ALU, in Abhängigkeit der zuletzt ausgeführten Rechenoperation, einzelne Bit(Flags) setzen kann. Auf die einzelnen Flags und ihre Bedeutung wird im Abschnitt der ALU näher eingegangen

9 Planung und Entwurf eines Prozessors

Der Inhalt der bisherigen Arbeit handelte von den Komponenten einer CPU und deren Funktionsweißen. Um den dargestellten Inhalt praktischer Vermitteln zu können, wird nun mittels einer Simulationssoftware eine CPU von Grund auf erstellt. Dieser Prozessor stellt keinen Vergleich zu modernen Prozessoren her. Er soll lediglich die Funktionsweiße der essentiellsten Bauteile beschreiben und einfache Operationen wie Sprünge und Subroutinen unterstützen.

9.1 Befehlsbreite

Am Anfang der Planung jeder CPU steht die Festlegung der benötigten Befehlsbreite. Je nachdem welche Features eingebaut werden sollen kann der Befehlssatz eingeteilt werden. Logisim bietet die Möglichkeit, einen 32-Bit Bus zu nuzten. Zu Erklärungszwecken werden die 32-Bit wie folgt aufgeteilt:

Tabelle 1: Befehlsbus

8-Bit	Opcode
8-Bit	Argument
16-Bit	Value

Opcode: Der Opcode beinhaltet den Befehl welche die CPU als nächstes Ausführen soll(z.B. MOV oder ADD). Es werden nicht mehr als 8-Bit benötigt, da nicht viele Befehle vorhanden sein müssen um die Basisfunktionalität einer CPU zu erzielen.

Argument: Das Argument wird nicht bei jedem Befehl verwendet. Diese 8-Bit sind eine Hilfestellung für Operationen bei denen eine genauere Spezifikation der zu ausführenden Tätigkeit benötigt wird. Beispielsweise wird bei der arithmetischen Operation ADD mit Hilfe des Argumentes angegeben, in welches Register das Ergebnis gespeichert werden soll.

Value: Die verbleibenden 16-Bit werden als Wertangabe benutzt. Durch diese 16-Bit wird gleichzeitig die Befehlsbusbreite innerhalb des Prozessors festgelegt, das heißt der Prozessor kann mit Zahlen arbeiten welche innerhalb der 16-Bit Grenze liegen (ohne

Vorzeichen maximal 65536). Einige Befehle in dieser CPU benötigen allerdings drei Parameter zur Ausführung. Um mit dem Argument drei Parameter bereitzustellen können die letzten 16-Bit in zwei 8-Bit Blöcke gespalten werden. Diese werden hier Quelle und Ziel genannt. Der Befehlssatz sieht bei diesen speziellen Befehlen folgendermaßen aus:

Tabelle 2: Befehlsbus mit drei Parametern

8-Bit	Opcode
8-Bit	Argument
8-Bit	Ziel
8-Bit	Quelle

Befehle, welche diese Aufteilung benötigen sind zum Beispiel ALU-Operationen oder der MOV Befehl, welcher den Wert eines Register in ein anderes schiebt.

9.2 Befehlssatz

Der Befehlssatz beschreibt die Befehle, welche die CPU ausführen kann.

Tabelle 3: Befehlssatz von VI-17

00000000	NOP
00000001	MOV
00000010	IN
00000011	STO
00000100	LEA
00000101	PUSH
00000110	POP
00000111	—
00001000	—
00001001	CALL
00001010	RETURN
00001011	ADD
00001100	SUB
00001101	INC
00001110	DEC
00001111	COMP
00010000	SHIFTL
00010001	SHIFTR
00010010	ROTL
00010011	ROTR
00010100	AND
00010101	OR
00010110	NOR
00010111	NAND
00011000	XOR
00011001	XNOR
00011010	JIT
00011011	JIF
00011100	JUMP

Die CPU soll die grundlegenden Aufgaben eines Prozessors erfüllen können. Die einzelnen Befehle des obigen Befehlssatzes werden nun kurz beschrieben.

00000000 NOP: No Operation. Es wird keine Operation ausgeführt.

00000001 MOV: Move. Überschreibt den Wert des Zielregisters mit dem Wert des Quellregisters.

9.3 Speicher

9.3.1 RAM/ROM

9.3.2 Stack

10 Implementierung einer Prozessorsimulation in Logisim

10.1 Logisim

Logisim ist ein Open Source Werkzeug für den Entwurf und die Simulation digitaler Schaltungen. Es bietet die Möglichkeit, größere Schaltungen aus kleineren Schaltungen herzustellen. Damit ist es möglich, ganze Prozessoren in Logisim zu entwerfen. Ein solch einfacher Prozessor soll nun im Folgenden implementiert werden.

10.2 Prozessor Komponenten

Der Prozessor besteht aus fünf Hauptkomponenten:

- Control Unit - Steuerungseinheit
- ALU - Arithmetisch Logische Einheit
- Registersatz
- RAM/Stack
- ROM

Control Unit - Steuerungseinheit: Die CU verarbeitet die Daten des Befehlsbusses und dekodiert die einzelnen Befehle, welche die CPU als nächstes ausführen muss. In Logisim wird der Befehlsbus mittels Komparatoren mit dem gesamtem Befehlssatz verglichen. Wenn ein Befehl gefunden wird sendet Logisim die notwendigen Steuersignale an die einzelnen Komponenten des Prozessors, um zum Beispiel die Register zum beschreiben freizuschalten.

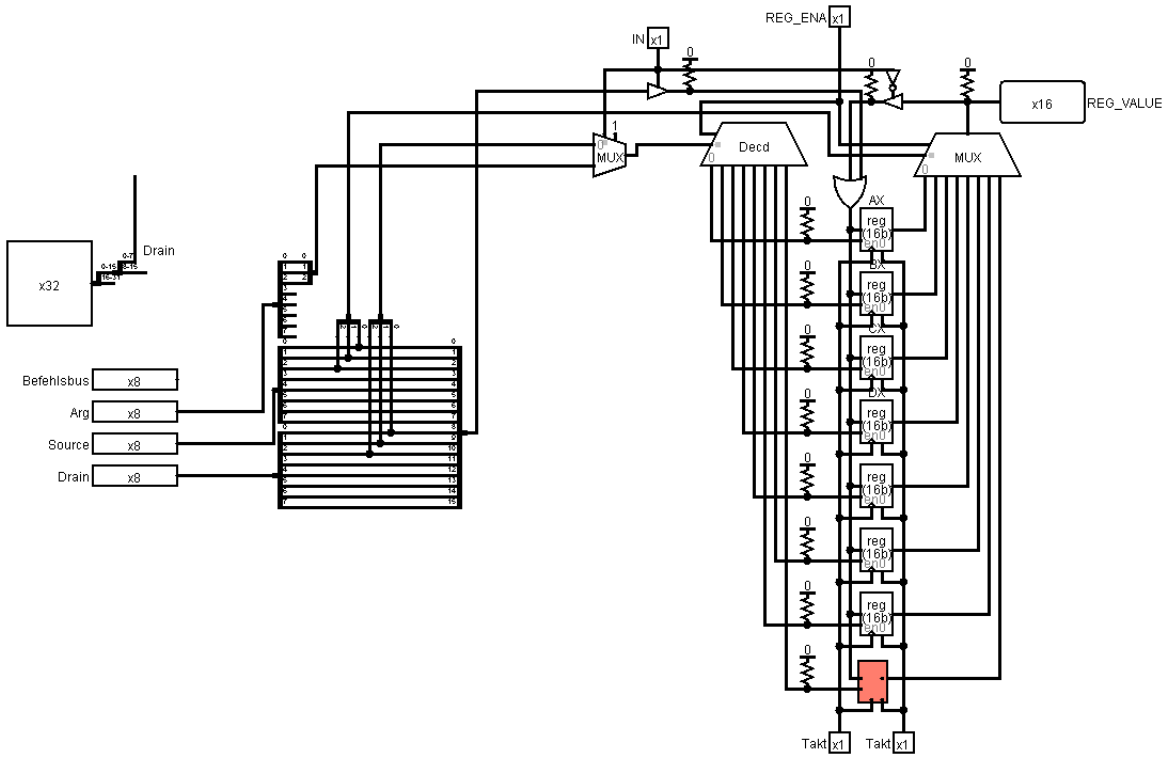

Abbildung 2: Darstellung des RegisterwerkTODO

10.3 Ausführung eines Assemblerprogrammes