Bau eines Platinen Computers

Alexander Wersching und Simon Walter

2021

Inhaltsverzeichnis

1 Anforderungen

2 Specification des Computers

Eine Computer hat eine Reihe Verschiedener high-level Aufgaben und Features die es ihm erlauben seiner Tätlichkeiten nach zu gehen! Diese Features können wie folgt auf gegliedert werden: Register, Memory, Banking, Input/Output, Flagen, Hardware Stack, Interrupts, Supervisor-Mode und das Instructuion Set.

Aber um zuerst ein paar standarts festzulegen: der Computer hatte eine Daten/Addresse Länge von 16 bit. D.h. er kann auf maximal 16 bit aufeinmal operieren bzw können maximal 16 bit an Daten/Addresse im Computer auf einmal bewegt werden.

2.1 Registers

Register sind kleine sehr schnelle Speichereinheiten, die genutzt werden können um zwischen Ergebnisse, local Variablen, argumente für Funktionen zu speichern oder um Daten im Computer zwischen den einzelnen Komponenten zuverschieben. Aus jedem Register kann *gelesen* werden (*Kopiert* des alten Wert an eine anderen Ort) oder *geschrieben* werden (den alten Wert mit einem neuen Wert *überschrieben*)

Dem Programmiere des Computers stehen bei uns 8 verschieden, jeweils 16 bit lange Register zur verfügung. Jedes Register hat dabei eine eigene Register Number (= RN). Machen Register erfühlen zudem noch speziell Aufgaben und sollten auch nur so verwendet werden. Diese Funktionen sind in Tabelle 1 zu sehen.

Tabelle 1: Register Aufteilung

RN	Name	Funktion
0	ZERO	Konstante 0x0000; Schreiben hat keine Effekt
1	IP	Instruction Pointer, zeigt auf akutuelle auszuführenden Instruktion
2	SP	Stack Pointer, zeigt auf obersten Element des Hardware Stacks, siehe 2.2.2
3	A	Alzweck Register ohne besonderen Aufgaben
4	B	...
5	C	...
6	D	...
7	FLAG	Enthält alle wichtigen Flaggen, siehe Unterunterabschnitt 2.1.1

Die Register A-B sind dabei die Register, dabei freie für jede nutzung. Es ist *stark zuempfelen* die spezieall Register, (ZERO, IP, SP und FLAG) *nicht* für die speicherung von Daten zuverwenden, da diese evt. Program abläufe stark störenen können. Sie sind ausdrücklich nur für ihre zugewiesenen Aufgaben da.

2.1.1 Flaggen

Das Register FLAG, enthält alle Status Flagen. Ein Flagge kann entweder gesetzt oder nicht gesetzt sein, daher kann jede Flagge als ein bit repräsentiert werden. Jedes Bit in FLAG stellt daher eine Flagge da. Flaggen werden bei unterschiedlichen Ereignissen geschaltet. Dies, und welche Flaggen es gibt ist in Tabelle Tabelle 2

Tabelle 2: Flaggen auf Teilung in FLAG

Bit	Name	Funktion	Schaltung
0	OF	Intager Overflow einer Rechenoperation an	Bei jeder ALU Operation
1	E	Gleichheit zweier Werte an	TEST arg[0] = arg[1]
2	G	Ungleicheit (>) zweier Werte an	TEST arg[0] > arg[1]
3	S	Ungleicheit (<) zweier Werte an	TEST arg[0] < arg[0]
4	P	IO-Geräte Eingesteckt siehe Unterabschnitt 2.3	TEST arg[0]
5	SUP	Supervisor-/Usermode	siehe Unterabschnitt 2.4
6			
7			
8			
9			
10			
11			
12			
13			
14			
15			

Alle Flagen die mit TEST als Schaltung gekenzeichnet werden, werden bei der Ausführung der TEST Operation geschaltet. Da die TEST Operation

Das FLAG Register verhält sich ebenfalls wie ein Register, daher können Flagge auch per Hand gesetzt oder nicht gesetzt werden. Eine Ausnahme macht SUP die nur manuel geschaltet werden kann wenn SUP gesetzt ist.

2.2 Memory

Währen entgegen die Register für local Variablen und zwischen Ergebnisse gedacht sind, ist Memory (= Arbeispeicher) gedacht um größer Datenmegne zuspeicher.

Er steuert sich dabei über LD und STR Instruktionen an. Die LD Instruktion kopiert eine Wert aus Memory und schreibt ihn in ein Register, währen STR an der angebenen Addresse den Wert mit dem einse Register überschreibt.

Memory ist in 65536 einzehlen und unabhänige Speicher zellen (Jede 16 bit große) auf gedeteilt. Jeder Zelle ist ein Address, welche von 0 bis 65535 reicht, zugeorndet. Kann Memory auf 65536words oder 131072bytes ($\approx$ 131.0kb) gleichzeitig zugreifen. [1]. Wichtig ist das die Gesamt Größe theorthisch weitaus größe sein kann, dank Banking siehe Unterunterabschnitt 2.2.1.

Dabei ist die Memory Karte in Abbildung 1 zu stehen. Die zwei Hardware Stacks sind dort gekenzeichnet mit ↑, mehr dazu in Unterunterabschnitt 2.2.2

Die mit Supervisor beschrieben bereich können nur dann gelesen oder geschreiben werden wenn die SUP Flagge gesetzt ist. Der Bereich der mit ROM versehen wurde, ist dabei nicht beschriebar, sondern kann nur gelesen werden. Eine STR Instruktion hat dabei die gleiche Wirkung wie ein NOP Instruktion

2.2.1 Banking

Bankend sind externe Memory Module die eingesteck werden können. Alle Banken sind gleich groß (18.4kw = 36.8kb) und der Computer unterstützt maximal 65536 unterschiedliche Banken. Jede Bank hat dabei in ID (BID), die von 0 bis 65535 reicht.

Der Computer hat zwei Bank Slots (Slot 1 `0x4000 - 0x3fff` und Slot 2 `0x4000 - 0x57ff`). Er kann nun auf diese Slots ein der theorthisch 65536 Bank legen [2]. Das Banken legen lässt sich über die SBK1 und SBK2, für Select Bank 1 und Select Bank 2, erreichen. Wenn man somit an diese entsprechende Memory Address schreibt, schreibt man in die Bank hinein

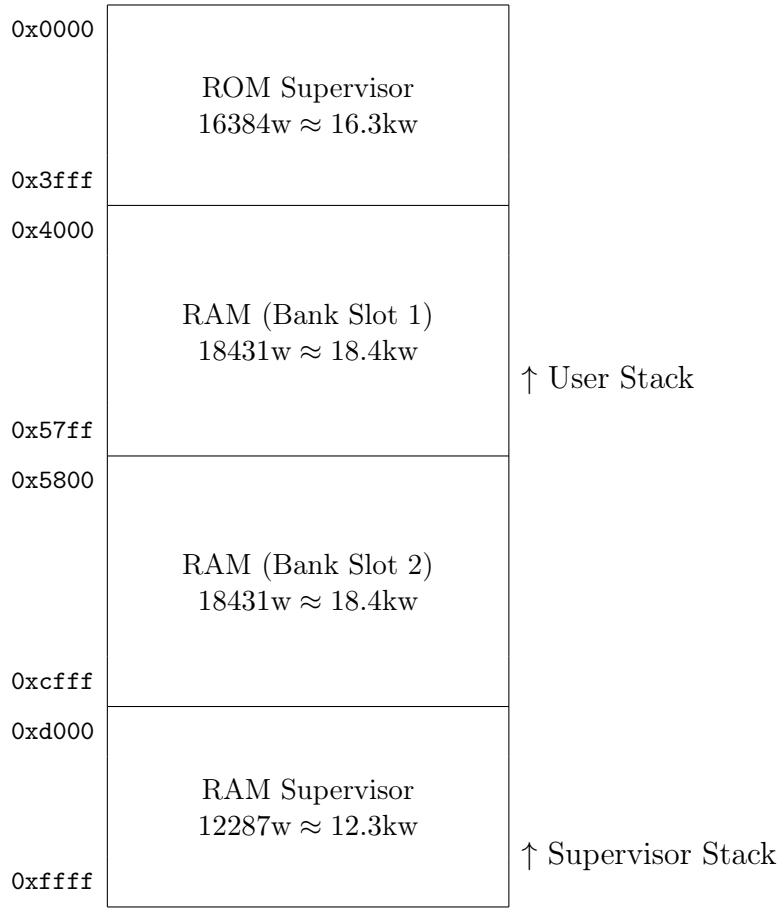
Durch dieses System lassen sich $65536 * 18.4\text{kw} = 1.2\text{Gw} = 2.4\text{Gb}$ ansprechen.

Das Bankensystem dient zudem dazu Prozesse von einander zu isolieren, da im Usermode der Computer nicht die Bank änderen kann, siehe Unterabschnitt 2.5.

[1] Die größe messen wir in words (w) oder in kilowords (kw), wobei 1word = 2bytes = 16bit

[2] Es ist dabei möglich die gleich Bank auf bei slots zu legen, auch wenn das nur bedingt nützlich ist

Abbildung 1: Memory Karte



2.2.2 Hardware Stack

Der Stack (oder auch Stapelspeicher) ist eine LIFO Datenstruktur. Sie dient hauptsächlich dazu Subrotinen als Funktion zuverwalten. Auf eine Stack kann entweder ein Wert drauf gelegt werden (PUSH) oder ein Wert oben abgehoben werden (POP).

Der Stack liegt dabei physische in Memory vor. Jedes Element des Stacks okupiert dabei eine Memory Zelle. Das Stack Point Register (SP) hält die Addresse zum oberste Wert des Stacks. Der Stack wächst dabei von hohen Addressen zu niedrigen Addressen. PUSH entspricht dabei der dektremtierung des Stack Pointers und POP der incrementierung.

Fundermental hat der Computer zwei Stacks. Der Supervisor Stack beginnt bei `0xffff` und wird genutzt wenn die SUP Flagge gesetzt ist. Der User Stack beginnt bei `0x57ff`, also am ende des Bank Slots 1 und wird genutzt wenn die SUP Flagge nicht gesetzt ist.

POP und PUSH schalten automatischen zwischen Supervisor Stack und User Stack bei Änderung der SUP Flagge.

2.3 Input/Output

Der Computer hat zudem die Möglichkeit mit verschieden IO-Geräte zu interagieren. Diese läuft über die sogenaten IO-Ports. Der Computer hat physische Slots wo IO-Geräte angesteckt werden können. Dabei bekommt jedes IO-Geräte eine eigene ID (IO-ID), welche im Bereich von 3 bis 65535 liegt. Die ersten drei IO-IDs sind für spezielle Aufgaben zugewiesenen. Die Interaktion funktioniert dabei wie mit Memory. Es kann zudem IO-Geräte Daten übertragen werden mit der OUT Instruktion und Daten vom IO-Gerät gelesen werden über IN Instruktion. Die IO-ID sind dann wie folgt vergeben, wie Tabelle 3 zeigt

Tabelle 3:

IO-ID	Funktion
0x0000	IO-Gerätetyp
0x0001	Interrupt Quelle
0x0002	Interrupt ID
...	
0xffff	physische IO-Geräte

Jedes IO-Geräte hat eine feste unveränderliche Typen nummer. Diese lassen sich über die IO-ID 0x0000 herausfinden. Diese sieht dann wie folgt aus:

```
OUT A 0x0000 ; A enthaelt die IO-ID fuer ein unbekanntes Geraet
IN  B 0x0000 ; B enthaelt nun die Typennummer des Geraets
```

2.4 Interrupts

Bei einem Interrupts (zu deutsch Unterbrechung) wird die Supervisor flag gesetzt und der Computer führt die Instruktion an der Stelle, die von IV (Interrupt Vector) festgelegt ist aus. Danach macht der Prozessor mit der gesetzt Supervisor Flagge weiter wie bisher.

IV ist dabei ein physischer Dipschalter am gehäuse des Computers. Es ist geraten ihn auf `0x3fff` einzustellen und dort eine CALL Instruktion zu positionieren, um IP zubewahren und Code zu verwalten des Interrups aus zufühen.

Interrups können durch folgen Ereignissen ausgelöst werden:

- IO-Geräte
- INT Instruktion
- Timer

2.5 Supervisor-/Usermode

Der Computer kann in ein einschrenckten Modus dem Usermode arbeiten oder in einem offene Modus dem Supervisormode, der vollkommen Zugriff auf die Hardware erlaubt. Der Computer ist daher so konzepiert das er einenen Supervisor hat, die anderen Prozesse started. Dieser soll folgen Aufgaben erfüllen:

Scheduling ist dazu gedacht, die limitierte Rechenzeit zwischen einzelnen Prozessen zu teilen. Um dies zureichen besitzt der Computer sogenten Timed-Interrupts.

Memory isolation zwischen Prozessen ist aus Sicherheitsgründen wichtig, um zu verhindern das sich fehlerhafte oder böswillige Prozesse, andere Prozesse nicht stören, diese wird durch Banking möglich gemacht.

Abstraction von der Hardware ist einweiter Grund. der Entwickler eines Programms soll nicht jeden Spezial Fall der Hardware configuration ein planen. Dies soll viel mehr der Supervisor machen, der über systemcalls (= Interrupts ausgelöst von Prozessen) ein Interface bieten.

Der Modus in welchen sich der Computer befindet wir durch die SUP Flagge gespeichert siehe Unterunterabschnitt 2.1.1. Eine gesetzte SUP Flagge entspricht Supervisor mode und eine nicht gesetzte Flagge dem Usermode. Selbige Flagge kann auch nur im Supervisormode verändert werden. D.h. aus dem Supervisormode kommt man sehr einfach in der Usermode, aber aus dem Usermode kommt man nur über Interrupts in den Supervisormode.

Der Computer startet im Supervisormode nach Poweron oder einem Reset.

Wie oben erwähnt ist der Usermode in folgenden Features einschreckt.

- Setzten der SUP Flagge
- Ausführung der IN/OUT Instruktion

- Zugriff auf Supervisor ROM und RAM
- Wechsel der Memory Banken
- Schalten der Interrupts

Wenn der Computer versuchen sollte einer dieser Aktionen im Usermode durch zuführen, hat diese die gleiche Wirkung wie ein NOP Instruktion.

2.6 Instructuion Set

3 Kommentar zur Specification

3.1 16 bit Daten/Address Länge

Die Address- und Datenlänge besagt viele Binäre Ziffern (bits) für eine Address und Daten verwendet werden. Je mehr Ziffern man verwendet desto größere Zahlen können in einer Operation verarbeitet werden wobei die maximal repräsentierbare Zahle $2^n - 1$ ist wenn n die Anzahl der Binary Ziffern (= Bits) ist und wenn man mit 0 anfängt zu zählen. Sie Tabelle 4

Ein eine n stellige Binär Zahl wobei n teilbar durch 8 ist, ist dabei relative angenehm, 8 bit = 1 byte. Das byte ist dabei die Basis Einheit für so gut wie alle informationstechnischen Standards. Daher ist es gute eine solches n zu wählen. Zudem kommen einige elementar ICs, z.B. Buffer oder Register, immer mit 8 bits.

Jedoch gilt je mehr Bits man verwendet, desto mehr Schaltung brauchen man, desto mehr Stromverbrauch. Da wir wie zwar nicht speziell auf Platz und Stromverbrauch optimieren, aber trozdem ein limitiertes Buget haben, können wir auch nicht eine belibig große Daten/Address Länge wählen.

Wir haben uns daher für 16 bit entschieden, der Codea man laut Tabelle 4 Zahlen bis zwischen 0 und 65535 darstellen kann. Diese Menge ist gerade große genug, um für die meisten Programme große genüge Zahl abzubilden [3].

Die Entscheidung zur größe der Daten/Address Länge ist eine sehr ausschlag gebende, daher haben wir diese zuerst festgelegt. Sie wird für viele weitere Entscheidung eine

[3] Für Programme die Zahlen welche > 65535 darstellen wohlen ist es ratsam immer zwei oder mehr Spreichereinheiten zunehmen oder bei Berechnungen, selbige auf zwei oder mehr Schritte auf zuteilen

wichtige Rolle spielen. Wichtig ist nur das wir maximal 65536 Zustände oder Zahlen von 0 bis 65535 darstellen können mit unseren 16 bit

Tabelle 4: Maximalerepersäntierbare Zahlen (startend mit 0)

n	$n^2 - 1$
2	3
4	15
8	255
16	65535
32	4294967295
64	18446744073709551615

3.2 Register Anzahl

Der Computer hat 8 Register. Wir haben uns für Acht ausfolgenden Gründen entschieden: Zu erst die minimale Anzahl anregisteren die wirklich gebraucht würde, ist 5! Man braucht mindestens zwei all Zweck Register um binäre Rechenoperation durch zu führen, einen Stackpointer, einen Instruktion Point und einen Ort wo man die Flaggen speichern kann [4].

Fünf lässt sich nicht mehr mit 2 bits repräsentieren ($2^2 = 4$), daher muss man 3 bit ($2^2 = 8$) nehmen. Der Grund warum wir nicht mehr genomen haben hat mit der Encodierung von Instruktionen zutun siehe Unterabschnitt 3.4. Somit haben sich 8 Register als das Optimum ergeben

[4] Theorthisch ist es möglich das Flaggen Register wegzulassen, wir haben uns aber für eine solches Register entschieden siehe Unterabschnitt 3.3

3.3 Aufbau von Conditionalen Operation

3.4 Alternative Instruktion Encodierung

3.5 Gedanken zum Supervisor/Usermode

3.5.1 Namensgebung

Ein Computer, der ein festes Programm hat ist zwar schon Gut, aber es geht auch besser. Moderne Computer besitzten zumeist ein Betriebsystem, Linux, MacOS, Android oder Windows. Wir haben uns entschieden unseren Computer so zubauen das er genau dies unterstützt. Wir haben uns aber entschieden das "Betriebsystem"nicht Betriebsystem sonder Supervisor zunehmen, da der Name Betriebsystem eine solche Funktionsvielfalt wie z.B. sie das Linux Kernal hat implizieren würde. Diese währe ein heilose Anmaßsung, der wir uns nicht schuldig machen wollten.

3.5.2 Gründe für die eingeschrenkten Features des Usermodes

Der Computer kann im Usermode einige Funktion nicht nutzen, welche im Supervisormode existieren und hier ist warum:

Setzten der SUP Flagge . Würde dies der Usermode können würde es das Konzept des User/Supervisormode obsolet machen. Der Prozess der sich im Usermode befänden könnte einfach das entsprechende Bit setzten und er hätte in der nächsten Instruktion die gleichen Rechte wie der Supervisor.

Ausführung der IN/OUT Instruktion . Dies ist hauptsächlich wegen der Abstraction der Hardware gedacht. Der Prozess soll nicht damit beschäftigt sein extra Code für die unterscheidlichsten Hardware Configurationen zu implentieren. Zudem kann es sein das bei Falscher benutzung von IO-Geräte physischer Schaden entsteht. Diese Maßnahme würde zumindest das Risiko verkleiner, da es nur noch ein Programm, der Supervisor, gibt wo die Komuninaktion mir IO-Geräten richtig implentiert werden muss.

Zugriff auf Supervisor ROM und RAM . Dies sollte eigentlich klar sein, dass es einem Prozess nicht gestated werden sollte mit den Daten des Supervisor direkt zu interagieren. Der Prozess könnte, wenn es diese Maßnahme nicht gäbe, sich so über das Überschreiben von Return-Addressen sich Supervisor-Rechte besorgen siehe Abschnitt 5.

Wechsel der Memory Banken . Die Idee hinter dem Banken ist nicht nur die erweiterung von Memory sonder auch die isolation von Prozess. Jeder Prozess lebt in seiner/seinen eigen Banken. Würde ein Prozess die Baken änderen können, so würde dieser mit dem Speicher anderer Prozesse interagieren und damit sicherheits kritsche Daten abgreifen oder mit der Ausführung dieser Prozesse stören können.

Schalten der Interrupts . Dies betrifft vorallem die Timed-Interrupts. Wenn ein Prozess im Usermode solche Interrupts setzten könnte, macht er damit das Konzept von Scheduling (siehe Unterunterabschnitt 3.5.3) obsolet. Der Prozess könnte einfach den Interrupt deaktivieren und er hätte beliege und unterbrochen Rechenzeit.

Muss dazusagen das diese sicherheits orientieren Überlegungen, nicht eintautoreffen wenn nur vertraunswürde Person an dem Computer arbeiten. Es kann aber auch sein das ein Prozess die angesprochen Störungen nicht mit absicht sondern durch schlecht programmieten Code hervorruft. Zudem können auch Prozesse selber Sicherheitslücken hervorrufen, was dazu führen kann, das ein Angreifer, erst ein Prozess im Usermode capert und dann mit sich Supervisormode beschaft. Und genau das wolle wir verhindern. Um den Schaden der entstehen kann nur auf ein Prozess zubeschrencken.

3.5.3 Schedling

Die Idee hinder Schedling ist sehr einfach. Der Supervisor lädt ein Programm in Memory, stellt all Banken soweit ein und aktiviert einen Timed-Interrupt, der nach einer bestimmten Anzahl an Instruktion auslöst. Der Supervisor springt nun in das Programm das er geladen hat und löscht die SUP Flagge. Der Computer befinden sich nun im Usermode und führt das Programm aus. Nach der vorher eingestellten Anzahl an Instruktion löst nun der Interrupt aus und der Computer Wechselt wieder in der Supervisor Mode, wo er alle Register etc. sichert und den nächsten Prozess started.

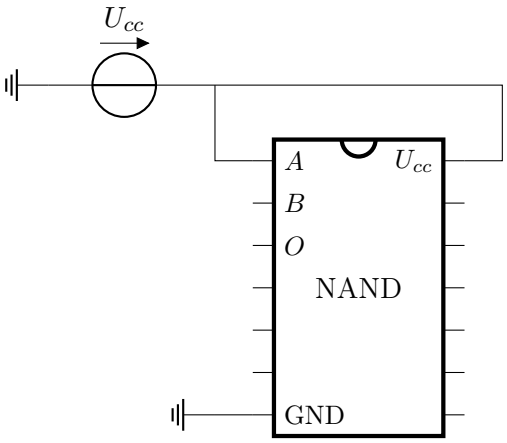
Diese Verfahren sorgt dafür das zwischen der Prozessen immer wieder hin und her gewechselt wird und so mehrer Prozesse zwar nicht parallel aber so das es nach außen, den Eindruck der Paraleliät erweckt.

4 Grundlegen Tests der Einzelnen Komponenten

4.1 NAND vs AND,OR,NOT

4.2 NANDs

Das NAND Gatter ist ein fundermentaller Logik Bautstein und bedarf daher einer Näheren Inspektion. Jedes NAND Gatte hat einen bestimmten propagation delay. Dieser gliedert sich einmal auf in High to Low (t_{PHL}) und Low to High (t_{PLH}). Diese ist den Datenblättern zu entnehmen. Wir haben uns dazu entschieden diese noch einemal selbst zutesten. Wir haben zu dem unterschiedliche NANDs von Unterschiedlichen Herstellern getested. Der Test Aufbau war dabei immer gleich:



Wir haben alle Test mir einer Basis Spannung von $U_{cc} = 5V$, $V_H = 5V$ und $V_L \leq 0.7V$ durch deführt

4.3 SRAM

Der RAM oder Random Access Memory ist der Hauptspeicher des Computers und bedarf daher ebenfalls einer genaueren Untesuchen. Er hat viele Funtionen die Getestet werden müssen:

1. Zeit zwischen OE und dem Erscheinen der Daten

2. Zeit zwischen Änderung der Addresse und dem Erscheinen der Daten

4.4 ROM

4.5 Buffer

Der Test Aufbau ist dabei wie folgt:

5 Security

6 Quellenverzeichnis