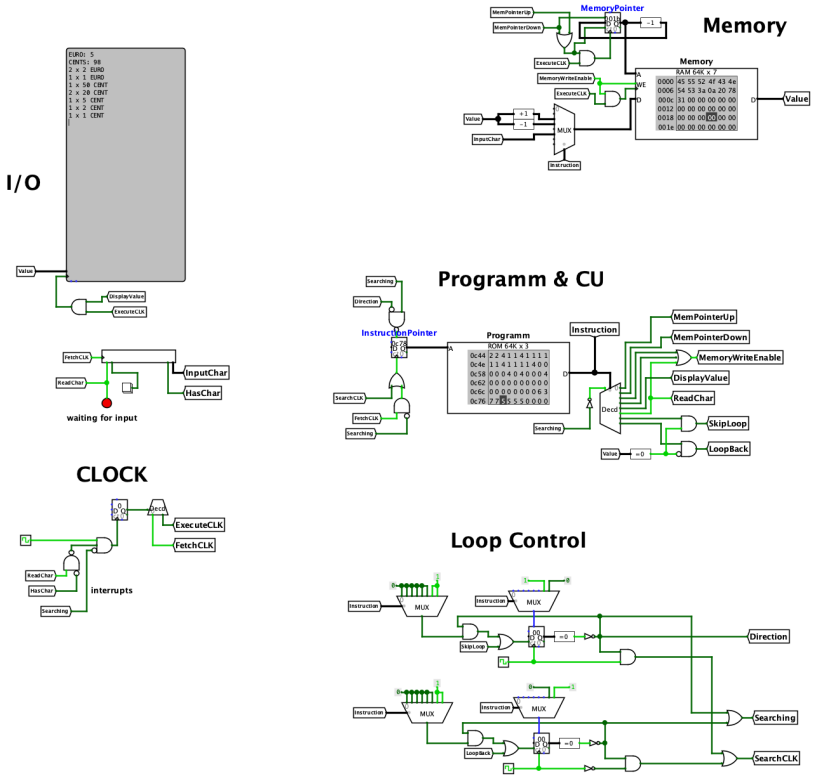
MPMP 2023

Jonathan Schulze 79886

CPU

```
Memory: 64k x 7 Bit
Programm: 64k x 3 Bit

Stages: 1. Fetch, 2. Execute
```



Instruktionen

Die esoterische Programmiersprache Brainfuck wurde gewählt, da sie wegen ihrer kleinen Anzahl von Instruktionen sowie dem Wegfallen von Argumenten nur eine einfache implementierte CPU benötigt.

Die Instruktionswörter sind 3 Bit breit, was alle 8 Instruktionen abdeckt.

- `0 000 >`
- `1 001 <`
- `2 010 +`
- `3 011 -`
- `4 100 .`
- `5 101 ,`
- `6 110 [`
- `7 111 ]`

Instruktionen `>` und `<` manipulieren den Memorypointer. Instuktionen `+` und `-` manipulieren den Wert, auf den der Memorypointer zeigt. `.` und `,` sind die Ein- und Ausgabeinstruktionen.

Logik kann mit den Schleifeninstruktionen `[` und `]` programmiert werden. Wenn bei der Instuktion `[` der Wert, auf den der Memorypointer zeigt, 0 ist, wird die passende schließende Klammer im Programmspeicher gefunden und das folgende Programm ausgeführt, ansonsten geht das Programm in die Schleife und springt beim Erreichen der schließenden Klammer wieder zur öffnenden Klammer zurück, wenn der Wert auf den der Memorypointer zeigt, nicht 0 ist. Dies ist in der CPU durch zwei Automaten, die bei Antreffen einer Klammer die CPU stoppen und mithilfe von Zählern durch das Programm iterieren, bis die schließende Klammer gefunden ist.

Speicher

7 Bit pro Speicherzelle wurden gewählt, um die Anbindung an die tty einfach zu halten. Diese kann nur ascii Zeichen von 7 Bit breite darstellen.

Programm

Generell können alle Brainfuck Programme, nachdem sie assembled wurden, ausgeführt werden. Der Assembler ist ebenso simpel wie die Sprache selber, optimierungen werden nicht durchgeführt. Es existieren bereits compiler die von unterschiedlichen Sprachen zu Brainfuck übersetzen, diese produzieren aber für die CPU zu große Programme. Das Programm euro.bf wurde von Hand geschrieben. Es gibt einem die minimale Anzahl an Münzen für einen 1-stelligen Geldbetrag an.

Programmablauf

1. init Chars

An Speicherpositionen 0 - 11 werden die ascii Zeichen `EUROCNTS:\n\sx` eingetragen.

2. ask for Euros
3. ask for Cents

An Speicherpositionen 13 - 18 werden die numerischen Werte Euro 10^0, Cents 10^1, Cents 10^0 sowie Flaggen für die Auswertung gespeichert.

4. calculate Euros
5. calculate Cents 10^1
6. calculate Cents 10^0

Die recursive Auswertung von den jeweiligen Beträgen ist eine 10-mal geschachtelte Schleife. Auf jeder Stufe wird der Betrag decrementiert. Wenn der Wert auf dieser Ebene 0 wird, ist die Höhe der Stufe, auf der dies passiert, mit dem ursprünglichen Wert gleich. Die Recursio wird durch das Setzen einer Flagge auf 0 gestoppt und die jeweiligen Münzen an die Stellen 21 - 28 aufaddiert.

7. print Coins

Die Anzahl an jeweiligen Münzen wird ausgegeben, wenn sie nicht 0 sind.