NOP
NIGHT OF PROJECTS

SHANGRILA

One Instruction Set Computer

NOP
NIGHT OF PROJECTS

Outline

- One Instruction Set Computer
  - Die Idee
  - Funktion
- Die Machine
- Shangrila VM
  - Interfaces
  - Tools
- Implementation
- Status & Zukunft

NOP
NIGHT OF PROJECTS

OISC >> Die Idee

- CPU mit nur einer Instruction
- CISC → RISC → OISC
  - Logische fortsetzung der Optimierung
- Turing-Complete
- Esotherisch / Akademisch

OISC >> Funktion

- Verschiedene Implementationen
  - SUBNEG
  - SUBLEQ
  - RSSB
  - MOVE

OISC >> Funktion

SUBLEQ A, B, C

- Speicher
  - für Daten
  - und Instruktionen
- Instruction Pointer
  - Beinhaltet Adresse des nächsten Befehls im Speicher

OISC >> Funktion

SUBLEQ A, B, C

- Subtract and branch if negative
- Alles nötige in einer Instruktion
  - Speicherzugriff / Manipulation
  - Rechenoperation
  - Bedingung
  - Sprung
- OPCODE kann entfallen

OISC >> Funktion

SUBLEQ A, B, C

- Adresse des Subtrahends
- Wird nicht verändert

OISC >> Funktion

SUBLEQ A, B, C

- Adresse des Minuends
- Adresse des Ergebniss
- Adresse des Wertes für Vergleich

OISC >> Funktion

SUBLEQ A, B, C

- Adresse des Sprungs
- Wird nicht verändert
- Optional - Default-Wert ist die nächste Instruktion

OISC >> Funktion

SUBLEQ A, B, C

- Wert an Adresse A wird gelesen
- Wert an Adresse B wird gelesen
- *A wird von *B abgezogen und Ergebniss wird in Adresse B geschrieben
- Vergelich von *B kleiner 0
- Wenn WAHR Sprung zu Adresse C

SUBLEQ A, B, C

```
*B = *B - *A
IF *B < 0 THEN
   GOTO C
ELSE
   GOTO NEXT
END IF
```

OISC >> Funktion

MOVE A, B

Kopiert Wert von Adresse A nach Adresse B

```
SUBLEQ B, B
SUBLEQ A, Z
SUBLEQ Z, B

SUBLEQ Z, Z
```

- B auf 0 setzen
- Z = -A
- B = 0 - Z

- Z = 0

OISC >> Funktion

SUBLEQ Z, Z, C

JMP C

Springt nach
Adresse C

Testet 0 - 0 <= 0

OISC >> Funktion

ADD A, B

Addiert Wert von Adresse A und Wert von Adresse B

```
SUBLEQ A, Z
SUBLEQ Z, B

SUBLEQ Z, Z
```

- Z = -A
- B = B - Z

- Z = 0

Die Maschine >> CPU

- Implementiert SUBLEQ

- 64 bit Wortbreite
- 64 bit ISP
- 64 bit Adressraum
- Instruktion durch 3 Worte

- Keine Addresierung unter Wortbreite

Die Maschine >> Memory

- 1-dimensionaler Adressraum
- Beinhaltet Instruktionen und Daten

- 64 bit Wortbreite / Adressraum

- Memory Mapping von Devices
  - Co-Prozessoren
  - Input / Output
  - Speicher
  - Andere Geräte

NIGHT OF PROJECTS

Shangrila VM

- CPU
  - Multi Core fähig
  - 1 Thread pro CPU
  - Halt und Stepping Flags
- Memory
  - Linearer Speicher durch Array
  - Laden von initialem Binary aus Datei
- Devices durch Plug-Ins
  - Memory Mapping

Shangrila VM >> Interfaces

- Plug-Ins für Devices
  - address_t getLength();
  - word_t peek(address_t offset);
  - word_t poke(address_t offset, word_t value);

Shangrila VM >> Interfaces

- Debug- und Steuer-Interface über Netzwerk
  - Pause und Stepping von CPUs
  - Events von CPU
  - Auslesen von Speicher
    - Werte
    - Events
  - Auslesen von Plugins
    - Werte
    - Events

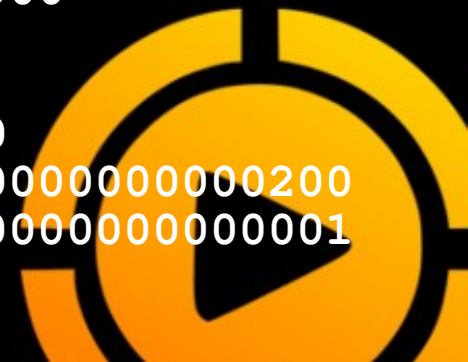
Shangrila VM >> Tools

- Assembler
  - Offsets
  - Labels
  - Macros
  - Jumps
- CLI Debuger
- GUI Debuger

NOP
NIGHT OF PROJECTS

Shangrila VM >> Tools

```
@ 0x0000000000000010

START:
# Clear temporary variables
  $ %A, %A
  $ %B, %B
  $ %C, %C

# Move Random value to %A
  $ %Z, %Z
  $ %RAND, %Z
  $ %Z, %A

# Move Random value to %B
  $ %Z, %Z
  $ %RAND, %Z
  $ %Z, %B

# Move values to math processor
# and add them together
  $ %Z, %Z
  $ %A, %MATH_A
  $ %B, %MATH_B
  $ %MATH_OP_ADD, %MATH_OP

# Move result to %C
  $ %MATH_R, %C

# Write %A, %B and %C to debug
output
  $ %A, %DEBUG
  $ %B, %DEBUG
  $ %C, %DEBUG

# Power of VM
  $ %POWER_OP_OFF, %POWER_OP


# Variables
@ 0x0000000000000060
A:   0x0000000000000000
B:   0x0000000000000000
C:   0x0000000000000000

# Constants
@ 0x0000000000000070
MATH_OP_ADD:  0x0000000000000200
POWER_OP_OFF: 0x0000000000000001
```



NIGHT OF PROJECTS

Implementation

- Implementation in C++
- Libev für Threads und Debug-Server
- Verwendet STL

Implementation >> Memory

- AdressMapper
  - Kennt alle Plugins
  - Leitet PEEK / P0KE auf Plug-In
- Memory
  - Grosses Array
  - Als Plugin implementiert
- Initialer Speicherinhalt aus Binary

NIGHT OF PROJECTS

Implementation >> CPU

```
instruction_t instruction;

// Get instruction from memory
instruction.a = this->addressMapper->peek(this->isp + 0);
instruction.b = this->addressMapper->peek(this->isp + 1);
instruction.c = this->addressMapper->peek(this->isp + 2);

// Get values referenced in instruction
word_t a = this->addressMapper->peek(instruction.a);
word_t b = this->addressMapper->peek(instruction.b);

// Calculate
word_t r = b - a;

// Write result back
this->addressMapper->poke(instruction.b, r);

// Jump depending on result
if (r <= 0)
  this->isp = instruction.c;

else
  this->isp += 3;
```

Status & Zukunft

- Status
  - 5000 Zeilen Code für eine Addition
  - Von Kongress zu Kongress
  - Debug- Client / Server in arbeit
- Zukunft
  - Compiler
  - FPGA
  - Betreibssystem

NOP
NIGHT OF PROJECTS

Dustin Frisch

dustin.frisch@gmail.com
http://dustin-frisch.de