F.A.E.E

Manual, Documentation and Technical Reference

Jene Litsch

Content

What is F.A.E.E?

F.A.E.E is an Emulator for a fictional 8-Bit CPU, written in C++17. It’s short for **F**ictional **A**ssembly **E**xecution **E**mulator. The project has no other real world purpose rather than being interesting and fun.

Hardware

The complete F.A.E.E machine contains:

- 8-Bit CPU @ ca. 4kHz default
- 32 KiB RAM
- 16 KiB ROM slot
- 256 IO device connections with 16x 8-Bit ports each

Start Parameter

The Executable allows to specify a few Parameter

Argv[1]	Filepath to Assembly Source Code [recommended: default = “”]
Argv[2]	CPU clock speed in Hertz as double/float negative results in default value [optional: default[-1] = 4096]
argv[3]	Log 0/else [optional: default = 0]
argv[4]	create Img 0/else [optional: default = 0]
argv[5]	.img file path [optional: default = “”]

Bsp:

```
./FictionalEmulator.exe ./programms/example.faee -1 1 1 ./imgs/testram.img
```

Instructions & Assembly Language

The Emulator uses its own Assembly Language.
Each Instruction is 4 Bytes in size, where the first Byte is the instruction code, and the other 3 are the parameters.
Unused Parameters will be ignored and non specified Parameters and Instruction-Codes are automatically set to 0x00/0.
The 16-Bit Memory Addresses are split into two Bytes and have to be written too in that way.
0x1234 → 0x12 0x34

All numbers can be written in Hexadecimal, Decimal and Binary

- 0x00 → 0xff
- 0 → 255
- 0b00000000 → 0b11111111

Example:

```
xmpl 0x42 0x3b 0xff
```

Below is a List with all instructions and their parameters.

System Instructions

Name	HexCode	Parmeter0	Paramter1	Paramter2	Description
exit	0x00	-	-	-	Ends the program and shuts down the CPU
setram	0x01	-	-	-	Sets CPU to RAM-Mode
setrom	0x02	-	-	-	Sets CPU to ROM-Mode

Memory Instructions

Name	HexCode	Parmeter0	Paramter1	Paramter2	Description
load	0x10	Target Register	Source RAM-Address 1/2	Source RAM-Address 2/2	Loads Byte from Memory to Register
write	0x11	Target RAM-Address 1/2	Target RAM-Address 2/2	Source Register	Writes Byte from Register to Memory
set	0x12	Target Register	Value	-	Writes value into register
copy	0x13	Target Register	Source Register	-	Copies the Content from one Register to another.
rload	0x14	Target Register	Source RAM-Address from Register 1/2	Source RAM-Address from Register 1/2	Loads Byte from Memory to Register
rwrite	0x15	Target RAM-Address from Register 1/2	Target RAM-Address from Register 2/2	Source Register	Writes Byte from Register to Memory

Signed Arithmetic

Name	HexCode	Parmeter0	Paramter1	Paramter2	Description
i_add	0x20	Result Register	Source Register A	Source Register B	Adds Register A and B A+B
i_sub	0x21	Result Register	Source Register A	Source Register B	Subtracts Register A by B A-B
i_mlt	0x22	Result Register	Source Register A	Source Register B	Mulitplies(A*B) Register A and B A*B
i_div	0x23	Result Register	Source Register A	Source Register B	Divides Register A by B A/B
i_mod	0x24	Result Register	Source Register A	Source Register B	Modulo Register A and B A%B
i_inc	0x25	Register			Increment Register
i_dec	0x26	Register			Decrement Register

Unsigned Arithmetic

Name	HexCode	Parmeter0	Paramter1	Paramter2	Description
u_add	0x30	Result Register	Source Register A	Source Register B	Adds Register A and B A+B
u_sub	0x31	Result Register	Source Register A	Source Register B	Subtracts Register A by B A−B
u_mlt	0x32	Result Register	Source Register A	Source Register B	Mulitplies(A*B) Register A and B A*B
u_div	0x33	Result Register	Source Register A	Source Register B	Divides Register A by B A/B
u_mod	0x34	Result Register	Source Register A	Source Register B	Modulo Register A and B A%B
u_inc	0x35	Register			Increment Register
u_dec	0x36	Register			Decrement Register

Bitwise Intructions

Name	HexCode	Parmeter0	Paramter1	Paramter2	Description
and	0x41	Result Register	Source Register A	Source Register B	Bitwise “and”
or	0x42	Result Register	Source Register A	Source Register B	Bitwise “or”
xor	0x43	Result Register	Source Register A	Source Register B	Bitwise “xor”
xnor	0x44	Result Register	Source Register A	Source Register B	Bitwise “xnor”
nand	0x45	Result Register	Source Register A	Source Register B	Bitwise “nand”
not	0x46	Result Register	Source Register A	Source Register B	Bitwise “not” Inverts 0s and 1s

Control Flow Instructions

Name	HexCode	Parmeter0	Paramter1	Paramter2	Description
ifnx	0x50	Register			Skips one command if Register is 0
ifjm	0x42	Register	Instr. Counter A	Instr. Counter B	If Register is 0 jumps to B else Jumps to A
goto	0x43	Instr. Counter			Jumps To Instructions

Logical Instructions for Signed Values

Name	HexCode	Parmeter0	Paramter1	Paramter2	Description
ieql	0x60	Register Out	Register InL	Register InR	Compares if values are equal true: 1 false: 0
ineq	0x61	Register Out	Register InL	Register InR	Compares if values are unequal true: 1 false: 0
ibigr	0x62	Register Out	Register InL	Register InR	Compares if R > L true: 1 false: 0
ismlr	0x63	Register Out	Register InL	Register InR	Compares if R < L true: 1 false: 0
ieqbigr	0x64	Register Out	Register InL	Register InR	Compares if R >= L true: 1 false: 0
ieqsmlr	0x65	Register Out	Register InL	Register InR	Compares if R <= L true: 1 false: 0

Logical Instructions for unsigned values

Name	HexCode	Parmeter0	Paramter1	Paramter2	Description
ueql	0x66	Register Out	Register InL	Register InR	Compares if values are equal true: 1 false: 0
uneq	0x67	Register Out	Register InL	Register InR	Compares if values are unequal true: 1 false: 0
ubigr	0x68	Register Out	Register InL	Register InR	Compares if R > L true: 1 false: 0
usmlr	0x69	Register Out	Register InL	Register InR	Compares if R < L true: 1 false: 0
ueqbigr	0x6a	Register Out	Register InL	Register InR	Compares if R >= L true: 1 false: 0
ueqsmlr	0x6b	Register Out	Register InL	Register InR	Compares if R <= L true: 1 false: 0

Memory Address Layout

16-Bit Address	Emulator Component
0x0000 - 0x7fff	RAM
0x8000 - 0xbfff	ROM
0xc000 - 0xefff	N/A
0xf000 - 0xffff	IO-Controller

IO-Devices

IO-Devices can be addressed exactly like Memory by using the load and write.

```
Input can be loaded like this: load 0x00 0xf_ 0x_P
     __    is the device ID [0x00 - 0xff]
     P     is the IO-Device Port [0x0 - 0xf]
```

Examples			
Device	Port	Load	Write
0x2a	0x8	load 0x00 0xf2 0xa8	write 0xf2 0xa8 0x00
0x42	0xf	load 0x00 0xf4 0x2f	write 0xf4 0x2f 0x00

There is are Standard I/O devices installed right out of the box.

Console		
The Console Outputs one Character at a time to c++ std::cout. Like all other IO-Devices the Console has 16 IO-Ports.		
Ports Out	0x0	Single ASCII Character
	0x1	Signed Integer
	0x2	Unsigned Integer
	0x3	Hexadecimal
	0x4	Binary
	0x5	N/A
	0x6	N/A
	0x7	N/A
	0x8	ASCII Character followed by a Linebreak [\n]
	0x9	Signed Integer followed by a Linebreak [\n]
	0xa	Unsigned Integer followed by a Linebreak [\n]
	0xb	Hexadecimal followed by a Linebreak [\n]
	0xc	Binary followed by a Linebreak [\n]
	0xd	N/A
	0xe	N/A
	0xf	N/A
Ports In	0x0-0xf	0