fabricOS Documentation

by click07

Installation

- Download Source from https://github.com/click07/fabric-os/fabricos.urcl
- Download a file of your liking from https://github.com/click07/fabric-os/
- Use Bram to Emulate https://bramotte.github.io/urcl-explorer

What's new in v0.3

- Rewrote File System to allow for fragmentation
- Inputting a directories name will change cwd if it is located in the current working directory (doesn't work with root yet)
- Added AND + NOR Instructions

FabricIL – Programming for fabricOS

⚠️ There isn't any compiler for fabricIL -> .bin yet, so all machine code needs to be entered manually.

Syntax is:
`0xGGHH 0xJJKK`
G is the first operand
H is the second operand
J is the third operand
K is the OPCode

So for Example:

```
IMM R2 7
IMM R1 3
ADD R3 R1 R2
SYS EXIT 1
```

would be:

```
0x0207 0x0002
0x0103 0x0002
0x0301 0x0203
0x0001 0x0001
```

To translate the HEX Values to a .bin file use a tool like
https://tomeko.net/online_tools/hex-to-file.php

File System

⚠️ Only files located in the root directory (Index: 0001) can be run as of right now

fabricOS uses a custom file system. The OS supports file types and custom file names. Reserved Values are therefor values that start with 0xFF. This FS allows for regular files to be used like directories while also containing instructions themselves. File Names are up to 4 characters long. Segments contain 32 words each. An implementation of this file system in hex would look something like this:

`FF00 0001` Type: Directory; Index: 1

`2F00` Name: “/”

`0000` Pointer to next segment: 0; Not a pointer

`FF01 0002` Type: Program; Index: 2

`FFFE 0001` Located in 1 (“/”)

`6578 6974` Name: “exit”

`0001` Pointer to next segment: 1

`FFFF FFFF` End of Header

`FFFF 0001` Segment: 1

`0000 0001`

`0000 0000`

`0000 0000`

`0000 0000`

`0000 0000`

`0000 0000`

`0000 0000`

`0000 0000`

`0000 0000`

`0000 0000`

`0000 0000`

`0000 0000`

`0000 0000`

`0000 0000`

`0000 0000`

`0000 0000`

`0000` Pointer to next segment: 0; Not a segment

File Types are:

⚠️ Only Directories, Programs and Packages can be run natively. All other file types require a program that can handle these types.

Name	HEX	Full Name	Description
DIR	0x00	Directory	Only a container for files. Inputting a directory name into console will set cwd
PRG	0x01	Program	Contains fabricIL. Inputting a program name into console will run the program.
PCK	0x02	Package	This file contains fabricIL but also other files for quicker access.
TXT	0x03	Text File	Contains Plain Text.

Instructions

⚠️ Instructions that are marked as gray are not implemented yet.

⚠️ This is incomplete.

OPCode	HEX	Name	Description	Operands
NOP	0x00	No OP	This does nothing.	
SYS	0x01	Syscall	Performs System Operation. Probably something I/O related.	SYSCALL, OP2, OP3
IMM	0x02	Immediate	Writes a value to a register.	REGISTER, VALUE
ADD	0x03	Addition	Adds two registers together, then stores it in a third.	DESTINATION, REGISTER1, REGISTER2
AND	0x04	And Operation	Performs AND Operation on two registers, then stores it in a third.	DESTINATION, REGISTER1, REGISTER2
NOR	0x05	Nor Operation	Performs NOR Operation on two registers, then stores it in a third.	DESTINATION, REGISTER1, REGISTER2
HLT	0x10	Halt	Halts the CPU.	

SYSCALLS

⚠ Syscalls that are marked as gray are not implemented yet.

OPCode	HEX	Name	Description	Operands
EXIT	0x00	Exit	Exits the system with a provided exit code.	EXIT CODE
SET	0x01	Set System Variable	Sets variable to the value of a register and stores it.	VARIABLE, REGISTER
GET	0x02	Get System Variable	Reads variable and stores it in a register.	VARIABLE, REGISTER
CHAR	0x03	Char I/O	Outputs char from to console or reads a char and stores it in a register. The Program is paused until the char is read.	IN/OUT (1/0), REGISTER
NUMB	0x04	Number I/O	Outputs a number to the console from a register. Input is yet to be implemented	IN/OUT, REGISTER
NEWL	0x05	New Line	Prints '\n' to the console	
SHPR	0x06	Shortcut Print	Prints a system String to the console.	REGISTER
GRPX	0x07	Graphics X Port	Basically a direct access to the Graphics Ports of URCL. OUT sets X Vertex, IN reads display width	IN/OUT (1/0), REGISTER
GRPY	0x08	Graphics Y Port	OUT sets Y Vertex, IN reads display height	REGISTER, IN/OUT (1/0)
COLR	0x09	Graphics Color	OUT sets pixel color at X and Y vertex, IN reads pixel color (RGBI or PICO8)	REGISTER, IN/OUT (1/0)

BUFF	0x0A	Graphics Buffer	OUT: 0 writes buffer to display, clears the buffer, and disables it, 1 enables writing to the buffer IN: Reads buffer state and stores it in register.	REGISTER, IN/OUT (1/0)
FILL	0x0B	Fill Screen with a Color	Fills the screen with a color. Add 01 to second operand for test screen	TEST MODE, COLOR

Planned: Note I/O, Input Devices

EXIT

When the system is halted, it will provide an exit code which is output to the console.
The meanings of the codes are as following:

0 - Planned Exit
The System was exited with a SYSCALL.

1 – Fatal Error
The System was exited as a result of an error. An example of this would be an invalid instruction or syscall.

2 – Unplanned Exit
The System was exited by a function or no exit code was given at HLT.

3 – No Storage File
The System failed to load due to a missing storage file.