Development Log

github.com/404dcd

2022-11-26

Sun May 1	**Add plan**
plan	completed first revision of plan
Sun May 15	**Getting started with ISA design**
bibliography	entered initial list of sources
spec - ISA	documented initial parameter types, registers
spec - CPU	virtual memory translation process
spec - OS	filesystem, extensive virtual memory paging details
Wed May 18	**Started work on opcodes and flags, do research**
bibliography	update sources
spec - ISA	core arithmetic, logical and branching instructions, arithmetic flags
Sat May 21	**Rework addressing modes and registers with the addition of width-changing prefixes**
spec - ISA	change parameter types to provide more useful options for iterating code, change registers to add more of them, added INC DEC SNX ZRX PUSH POP, added JXXX based on flag status, prefixes (to replace 8b 16b registers, which is a better way to do it on reflection)
Sun May 22	**Add calling conventions and stack layout**
spec - ISA	added PUSHR POPR CALL, added ZRF NGF flags
spec - CPU	calling conventions for functions, function stack layout - this is mainly assembler details
Sun May 29	**Finish detailing all instructions, add a couple**
spec - ISA	added JUMP ASL RET NOP, completed first revision of detailed instruction behaviour (worded explanations, in addition to opcode table)
Tue May 31	**More research**
bibliography	update sources
Sat Jun 4	**Tidies flags, adds special registers and instructions for them**
plan	make plan more detailed by adding tasks for kernel features memory, keyboard, screen, disk, process creation
spec - ISA	add special registers, add CPFLGR CPIVTR WRIVTR WRPDBR for those registers, add flags for those modes too (IEF, VMF)
Sun Jun 5	**Adds some more instructions for ports and interrupts**
spec - ISA	add SETIEF CLRIEF SETVMF CLRVMF for new flags enabling modes, add and explain INP OUT GENINT IRET LMA
Mon Jun 6	**Details interrupts**
spec - CPU	actions processor takes when interrupt is triggered, initial table of exceptions and interrupts from devices (this gets changed later)
Fri Jun 10	**Details ports, lists IO devices, a synopsis for the memory controller**
spec - CPU	list and detail hardware device ports, more memory controller behaviour

Sat Jul 16	**first go at emulator but Rust is too restrictive, switching to C**
emulator_v1	initial and only commit for Rust attempt (got as far as reading instructions in, beginning loop and interpreting the opcode but found borrowing rules too strict and unworkable)
Fri Jul 22	**emulator sprint (over past 3 days)**
main.c	virtual memory translation mode, full parsing of instructions, execution for: flag setters, NOP, arithmetic instructions but not shifts, special register getting/setting operations. On the to-do list: PUSHes and POPs, returns, shifts, branches, ports, CALL, GENINT, interrupts
Sat Jul 23	**basic emulator mostly done**
main.c	bug fixes, execution for: PUSHes and POPs, returns, shifts, branches, CALL
Sun Jul 24	**emulator and assembler running**
assemble.py	implement the whole assembler except for strings, loading address etc directives - those coming later
main.c	bug fixes (needed to add brackets for C's operation associativity)
Thu Jul 28	**updates spec for what's been implemented**
main.c	bug fixes, emulator now gives cycles executed stat at end
plan	modify plan to document switching to C
spec - CPU	swap source and destination order in instructions (this is more human-readable), some minor *exceptions* to instruction restrictions (exceptions to destination not being immediate, a max one memory reference) added for specific instructions
Fri Jul 29	**gets screen working**
assemble.py	bug fixes
main.c	start implementing OUT, spawn a window runner in another thread
window.c	code to create X11 window, run its event loop, do direct memory access (transferring data) from CPU memory, but not yet handle keystrokes
Tue Aug 2	**serial port, printing strings and numbers, implementing interrupts**
assemble.py	bug fix, add directive for storing strings (realised it would be a nice to have)
_fn_out.txt	function fn_out_str, function fn_out_uint (marks the start of the kernel/OS)
main.c	some interrupt handling code (not complete), serial OUT text working, emulator execution for GENINT
window.c	blank the screen until actually drawn to
spec - CPU	remove double fault and tick exception/interrupt, use more relevant interrupts instead such as an address too large, continue to document how interrupts work, document serial out and display screen
Sat Aug 6	**added go-style channels, interrupts working, IO ports working, keyboard implemented**
channel.c	implemented handy go-style channels for use in emulator
main.c	set up channels for keyboard and display IO, finish interrupt implementation, execution for INP
window.c	sending keystrokes to main loop as interrupts
spec - CPU	some minor adjustments for port buffers being modelled by channels, document the updated screen resolution

Tue Aug 9	**proper ROM → BOOT → kernel loading, cleverer assembler, disk image creator**
assemble.py	implement loading address directive and file inclusion, assembling multiple files
channel.c	add blocking on channel until value ready
disk.c	implement the whole disk device
generate_disk.py	implement tool to build disk image from directory of binary files
main.c	run disk device as well, HLT now waits for interrupt if enabled rather than quitting, OUT finished
BOOT.txt	bootloader that reads file system table, finds kernel, loads and jumps to it
ROM.txt	ROM that loads first sector of disk and jumps to it
_fn_str.txt	implement function fn_strcmp, compares two strings
kernel.txt	now decided syscalls. begin planning areas for data store
_handlers.txt	very simple keyboard handler, just prints to serial
Wed Aug 10	**setting up page tables, turning on VM, kernel relocates to higher half**
main.c	remove source type constraints on shifting operations
_fn_freephys.txt	implement function fn_freephys_reserve, scans for free physical page from indicator bit-field
kernel.txt	code to set up identity paging, turn on VM, then make the actual maps. not properly jumping there yet
spec - ISA	remove source type constraints on shifts (they were just unnecessary and made shift instructions relatively useless)
Fri Aug 12	**more mapping of memory kernel needs, first syscall**
assemble.py	add relative loading address directive
main.c	bug fix
_fn_map.txt	implement function fn_map_man to add entries into page tables, fn_map_auto calls fn_map_man with an automatically found free page
_handlers.txt	default handlers for all exceptions
_syscalls.txt	implement syscall map
kernel.txt	jump to higher half now, mapping some data storage pages, registering handlers and syscalls
Fri Aug 19	**syscalls unmap, exit, spawn plus other tidy**
_fn_disk.txt	implement functions fn_disk_find, fn_disk_read
_fn_freephys.txt	implement function fn_freephys_makeavail, marks a physical address as available to use
_fn_map.txt	implement function fn_map_free, removes a page mapping
_fn_zeropage.txt	implement function fn_zeropage, zeroes a given page
_handlers.txt	optimisation of default exception handlers that now kill the offending process, write handler for disk
_syscalls.txt	implement syscalls unmap, exit, spawn
kernel.txt	more allocations of storage space, initialising memory for syscalls, etc
renaming.txt	create list of instructions to be renamed, so that they more accurately reflect their function
Wed Aug 24	**fix spawn and exit, load init, syscalls alloc free flen**
generate_disk.py	bug fix
_fn_disk.txt	fix mis-implementation in fn_disk_read
_handlers.txt	bug fix
_syscalls.txt	bug fixing, implement syscalls alloc, free, flen, make a start on fread
init.txt	hello world, this is a template for user mode process
kernel.txt	more allocations of storage space + kernel spawns the init process now

Fri Aug 26	**syscall fread, also working on custom font in photoshop**
font.bmp	begin work on font
_syscalls.txt	finish implementing syscall fread
kernel.txt	continue to add required setup for newly implemented syscalls
Mon Aug 29	**added kernel display text driver**
font.bmp	font completed
_fn_display.txt	implement the whole ASCII text driver (several functions)
_fn_map.txt	add function fn_map_translate
kernel.txt	load font, other setup for running of text driver
Thu Sep 1	**keypresses are translated to ASCII and drawn, fixed syscall fread and made print and get**
generate_keymap.py	this tool generates a keymap file
_fn_display.txt	scroll screen lines (bubble) if the next line would go off the screen, bug fixes
_fn_str.txt	implement function fn_str_fromint, generate string representation of integer
_handlers.txt	keypresses are now drawn to the screen
_syscalls.txt	bug fixes, implement syscalls print and get
init.txt	simple echo program to test new syscalls
kernel.txt	cleaning up, load the keymap
Mon Sep 5	**Fixed syscall get and malloc, everything seems to work now**
_syscalls.txt	major bug fixing
Sun Sep 11	**Renaming, Prime finder in userspace**
_fn_out.txt	bug fixes
_fn_str.txt	bug fixes
init.txt	implement assembly program to find prime numbers less than the typed number, as the demo
kernel.txt	allocates a separate "string creation space", which fixes some bugs
**.txt*	rename instructions following renaming.txt
Wed Sep 14	**Completing various to-dos, including writing some specification. Fixed a syscall**
_syscalls.txt	bug fix, do the to-dos
spec - CPU	document and specify disk device, keyboard device, initial ROM execution
Mon Sep 19	**Document and specify kernel behaviour**
spec - OS	document bootloader, kernel startup tasks, most syscalls
Thu Sep 22	**Finish documenting syscalls, begin documenting assembler and emulator**
spec - OS	document + specify remaining syscalls, assembly language syntax and start documenting the assembler tool
Sat Sep 24	**Artefact finished**
spec - OS	finish documenting assembler, completely document emulator, a finalised specification is complete