Сводная инфа по Sprinter`у.

1. Историческая справка.

Спринтер 2000 разрабатывался в конце 90х – начале 2000х Иваном Макарченко (Ivan Mak, позднее Winglion, умер 9 апреля 2013го года).
Продавался Питерской фирмой «Петерс Плюс». С 2005го года Петерс Плюс сняла с производства и продажи все модели Спринтера и покинула платформу zx (чуть позднее и вовсе закрылась). В 2007м году Иван начал постепенно открывать тему Спринтера, а в 2009 году выложил в свободный доступ все имеющиеся материалы (включая исходники ядра, схему и файлы для производства).

На данный момент в массах находятся 3 версии – Sp2000, Sp2002s и Sp2003. Соотношение распространённости мне не известно. Отличаются между собой видеопамятью. На моделях Sprinter 2000 установлена видеопамять W24512AK-15 в DIP корпусах. На моделях Sprinter 2002s и Sprinter 2003 установлена видеопамять AS7C1024-12JC в SOJ корпусах. При этом, на Sp2002s память установлена «квадратом» (4 микросхемы), а на Sp2003 в шахматном порядке (так же 4 микросхемы). Однако, есть рабочий экземпляр (можно сказать, что это Sp2016) с работающей видеопамятью **W24010AS-35**.

Последняя версия биоса в собранном виде – 3.04. Стабильной считаются 3.03 и 3.04 (предположительно, для Sp2002s - 3.03, а для Sp2003 нужна 3.04). Различия в этих биосах только в ядре (fpga). Скорей всего, разница там в таймингах. Примечательно то, что на Sp2002s и Sp2003, если прошить биос ниже 3.03, то можно наблюдать периодические (или даже постоянные) артефакты на экране. При этом на Sp2016 (будем этот экземпляр так называть), таковых проблем не обнаружено.

2. Некоторые особенности машины.

Процессор работает на частоте 21мгц в режиме Спринтера. В режиме Спектрума имеет 3.5мгц с турбированием до 7мгц. Графические режимы – 320x256x8, 640x256x4, 256x192 (zx), text mode 80x32. Звук – AY (не железный, а эмуляция средствами fpga), covox/covox-blaster. Весь звук выводится через TDA1543. Имеется HDD. FDD может работать с дисками высокой плотности. Клавиатура – можно подключать PS/2 клавиатуры (если вместо DIN5 запаять ПСный разъём). Мышь – последовательная (проще говоря – компортовая или serial mouse).

Сразу хочу отметить, что установленный процессор имеет несколько своих собственных портов и некоторые из них пересекаются с имеющимися у Спектрума. Например, порт 0x1f конфликтует с внутренним портом процессора. В режиме Спринтера для обращения к этому порту следует обращаться к порту 0x0f, а в режиме Спектрума обращение к 0x1f перехватывается со стороны fpga и направляется на 0x0f. Список всех вн.портов процессора вы можете найти в документации на процессор. На данный момент мне известны следующие порты: 0x10 - 0x1F, 0xEE, 0xEF, 0xF0, 0xF1, 0xF4.

Covox и Covox-Blaster это одно и тоже устройство работающее в разных режимах – обычный covox как у пентагона и режим covox-blaster, имеющий дополнительный fifo. Кроме этого, доступны режимы – моно. Стерео, 8бит, 16бит, частоты от 8кгц до 115кгц (но по факту выше 44кгц нет смысла использовать).

AY в Спринтере представлен в виде эмуляции на ahdl и он является частью ядра. Многие могут подумать, что раз это эмуляция, то значит «фи». Но могу сказать, что эмуляция тут очень и очень качественная. Не каждый сможет отличить на слух местную эмуляцию от настоящего AY. Работает во всех режимах.

Для переключения машины в режим Спектрума нужно использовать программу – spectrum.exe. У программы есть свои «ключи» для запуска (прочтите мануал к программе), которые можно записать в виде конфигурационных файлов. Приведу стандартный конфиг для запуска Спринтера в режим обычного Пентагона 128:

```
Pentagon128
C:\zx\sp_128.bin
C:\zx\sp__48.bin
C:\zx\sp_trd.bin
C:\zx\sp_exp.bin
C:\zx\sp_exp.bin
C:\zx\sp_exp2.bin
/7ffd /to-trdos /ret-fn
```

Можно добавить ключ /turbo, тогда по кнопке F12 можно включать или выключать турбирование (при запуске с этим ключём турбирование включено сразу).
Могу сказать, что Пентагон «показывается» тут весьма точно. Проверить можно на известной демке **Rage** от X-Trade. Известный эффект с «крутящейся» радугой на весь экран (включая бордюр) тут показывается без искажений.

Спринтер имеет акселератор для пересылки данных. Данные можно кидать из озу в озу, из озу в экран, из экрана в озу. Пересылать данные между озу и портами или fast-ram на данный момент нельзя. Максимальный размер одной транзакции 256 байт. Минимальный 1 байт. Нужно иметь ввиду, что размер транзакции равный 0, будет равен 256 байтам. Таким образом, если вы указываете размер 0xff, это это будет 255 байт, а 0 это будет 256 байт. Кроме пересылки акселератор может применять заполнение области байтом, применять операции or, and и xor на пересылаемые данные. Ожидать окончания пересылки или заполнения не требуется. Акселератор работает через перехват команд процессора.

Самой главной фишкой Спринтера является то, что он может применять различные конфиги на лету. Однако именно этот механизм мне на данный момент не понятен. Ближайшим примером по смене конфига на ходу является демка Дума от Ивана, а так же игра TITD в которой почти полностью меняется конфигурация машины. Однако, как я понял из различных источников (включая некоторые исходники), смена конфигурации тесно связано с программной перезагрузкой машины. Пока точно не понял, но вероятно, что до того как посылать байт ребута, требуется загрузить в страницу памяти не выше 127й нужный конфиг, разместить нужный код в нужных страницах и передать байт ребута в нужный порт или страницу. Произойдёт перезапуск машины, но вместо биоса будет производиться запуск вашего кода. При этом, файл конфигурации хранится в неизвестном для меня формате и он нигде не описан. Есть предположение, что он чем-то пожат (т.к. полный конфиг всей машины занимает около 59кб, а тут речь идёт только о 16кб, таким образом, тут либо не полная смена конфигурации либо файл чем-то пожат, а процедура пзу при перезапуске распаковывает данные и кидает в плисину). Тут нужно ещё дальше разбираться.

Спринтер имеет несколько графических режимов. Для каждого может быть использована палитра 24 бита (16млн. цветов). Вообще говоря, палитра задаётся в формате BGRi, т.е. 32 бита, однако младшие 8 бит (I - ignored) всегда игнорируются.
Каждый графический режим имеет по 2 экрана (аналогично экрану Спектрума 128). Т.е. это основной (видимый) экран и скрытый.

3. Программирование

Начну с того, что все программы для Спринтера лучше всего писать под его ДОС. Некоторые вещи быстрее и/или удобнее выполнять именно под ДОС, чем вручную. У каждого исполняемого файла (EXE) есть свой заголовок:

```
                org 0x8100-512
start_addr:
                DB 'EXE'                ; EXE ID
                DB 0            ; EXE ВЕРСИЯ
                DW 0x200        ; 512, МЛ. СМЕЩ. КОДА
                DW 0            ; СТ. СМЕЩ. КОДА
                DW 0            ; END-BEG, ПЕРВИЧНЫЙ ЗАГРУЗЧИК
                DW 0,0          ; РЕЗЕРВ0
                DW 0            ;
                DW begin        ; АДРЕС ЗАГРУЗКИ КОДА
                DW begin        ; АДРЕС ПЕРЕДАЧИ УПРАВЛЕНИЯ
                DW 0xBFFF       ; АДРЕС СТЕКА
                DS 490          ; РЕЗЕРВ1
```

Вам важно запомнить, что такой (подобный) вид заголовка годится для небольших проектов/программ, размер которых не превышает ~32кб. Весь заголовок имеет размер 512 байт. РЕЗЕРВ1 добивает заголовок до конца и уже на 513м байте начинается код программы. Однако, ДОС позволяет запускать большие программы, не дробя их на куски. Примером такого проекта является, игра TITD (или его инсталлер, который имеет размер 1мб). Чтобы ДОС мог грузить большие программы, в заголовке требуется поменять несколько моментов:

```
                org 0x8100-0x16
EXEHeader:
                db "EXE"
                db 0
                dw LoaderStart -EXEHeader       ;Размер заголовка
                dw 0
                dw LoaderEnd - LoaderStart      ;Размер загрузчика
                dw 0,0,0
                dw LoaderStart                  ; Адрес загрузки (загрузчика)
                dw LoaderStart                  ; Адрес передачи управления
                dw LoaderStart -1               ; Стэк.

LoaderStart
                Тело загрузчика…
```

Как можно заметить, в заголовке теперь отсутствует ds 490, а так же поменялись некоторые поля в заголовке. Тут, в общем-то, всё более или менее понятно и логично – размер заголовка, размер загрузчика, адрес загрузки, передачи и стэк. Там где было ds 490, теперь должен быть код загрузчика, который догрузит остальные части программы, раскидает их по страницам памяти (если нужно) и в конце сделать jp mainStart.
Для себя я пока делаю для большинства программ вот так:
Выкинул заголовок (по первому варианту) в отдельный файл head.inc, в теле основной программы (например, main.asm) делаю так:

```
                device zxspectrum128

                include "..\..\include\dss.inc"
                include "..\..\include\head.inc"

begin:          бла бла бла...код программы


code_end:
                savebin "progname.exe", code_start,code_end- code_start
```

Содержимое head.inc

```
                org 0x8100-512
code_start:
                db "EXE"
                db 0
                dw 0x200
                dw 0
                dw 0
                dw 0
                dw 0
                dw 0
                dw begin
                dw begin
                dw 0xbfff
                ds 490
```

Сборка осуществляется через батник:

```
@echo off

if EXIST progname.exe (
        del progname.exe
)
if EXIST progname.lst (
        del progname.lst
)

..\..\asm\sjasmplus.exe main.asm --lst= progname.lst
if errorlevel 1 goto ERR
echo Ok!
goto END

:ERR
del progname.exe
pause
echo ошибки компиляции...
:END
```

Для работы с ДОСом есть отдельный мануал, когда-то написанный Иваном. В нём приведён список системных вызовов и аргументы к ним (с редкими комментариями). В целом, по этой документации можно понять всё. Там описаны вызовы для работы с файлами, памятью, печать

символов, переключение в графические режимы и т.д. Тут я дам несколько комментариев к некоторым вызовам.

Первое. Все номера всех вызовов в ДОСе всегда передаются через регистр “**c**”.
Второе. В отличие от ТРДОС, тут не стоит забирать какую угодно страницу памяти. Это может привести к непредсказуемым последствиям. Например, когда запускается дос и следом консоль, консоль получает для себя некий кусок памяти (1 или 2 страницы). Какие это страницы – не известно. Запустили Flex Navigator – аналогично – страницы будут для него выделены динамически. Заранее не известно, какие это страницы. Если вы будете принудительно забирать под себя страницы памяти, не «информируя» об этом систему, то вы рискуете либо, затереть данные от коммандера, консоли, доса или ещё какой программы либо, другая программа затрёт ваши данные, если они должны сохраняться в памяти. Для выделения памяти у ДОСа есть соответствующий вызов – getmem (0x3d). Дос может обрабатывать запрос сразу в несколько страниц. Запросив, к примеру, 8 страниц ДОС возвращает вам в регистре «**A»** идентификатор. Этот идентификатор необходимо передать в биос, указав в регистровой паре HL адрес сохранения таблички с выделенными страницами. Нужно помнить, что эта табличка должна быть на 1 байт больше, чем количество запрашиваемых страниц. Последний байт в этой табличке будет равен 0xFF как признак конца этой таблички. Вы можете поэкспериментировать с этим вызовом и проверить результаты либо в отладчике эмулятора либо, в отладчике demon.exe на реальной машине (demon.exe, прочтите документацию на этот отладчик). Пример запроса 8 страниц памяти

```
                ld c,getmem
                ld b,8
                rst 0x10                ;на выходе A содержит идентификатор выделенных страниц
                ld hl,pages
                ld c,0xc5
                rst 8
                …

Pages:          ds 9            ;9й байт будет иметь значение 0xff после rst 8, а первый байт можно
                                ;использовать как идентификатор
```

Чтобы подключить нужную страницу потом, можно использовать вариант:

```
                pop hl          ;восстановили pages
                ld a,(hl)
                out (cpu_w3),a ;подключили страницу в 3е окно проца.
```

И так далее и тому подобное. Изобретайте свои конструкции, проверяйте и эксперементируйте.

Теперь хочу прокомментировать работу с акселератором. **Перед тем, как использовать акселератор, запретите прерывание!!!**
Акселератор включается путём перехвата команд процессора. Есть основной набор команд, которые перехватываются и распознаются ядром как обращение к акселератору: LD A,A; LD B,B; LD C,C; LD D,D; LD E,E; LD H,H, LD L,L.

Назначение команд следующее:

LD B,B - выключить акселератор.

LD D,D - включить акселератор в режим приема байта размера блока
далее следует команда типа LD A,cnt, где cnt и будет новым
размером блока. Если размер блока был установлен ранее,
его можно не устанавливать.

LD C,C - Операция Fill - заполнение одним байтом. Последующая команда типа LD (HL),A приведет к заполнению указанного ранее количества байт значением A

LD E,E - Операция Fill для графического экрана - заполнение вертикальных линий.

LD H,H - rezerved

LD L,L - копирование блока. Последующая команда типа LD A,(HL) приведет к заполнению ОЗУ акселератора данными из адреса (HL), а команда типа LD (DE),A приведет к перезаписи данных из ОЗУ акселератора в основное или видео-ОЗУ.

LD A,A - копирование блока для графического экрана подобна команде LD L,L, но работает с вертикальными линиями экрана.

Самый простой пример по переброске куска данных при помощи акселератора:

```
di
ld d,d
ld a,0
ld l,l
ld a,(hl)          ;в этот момент данные залетают из озу во вн.память fpga
ld (de),a          ;а тут наоборот данные залетают в озу.
ld b,b
```

Как можно заметить, в примере и в описании сказано, что размер транзакции может быть задан, якобы, только через конструкцию вида ld a,cnt. На самом деле это не так. Во1х, вместо регистра «А» можно использовать любой другой по вашему усмотрению. Во2х, применятся, могут и иные конструкции, вроде pop af или подобные.