манускрипт основан на работах некоего si17911, поэтому некоторые части не переведены (на английском они пон¤тнее).

sjћя"№ PSX

X0x00000000-0x001FFFFF - RAM
X0x1F000000-0x1F?????? - параллельный порт (PIO)
X0x1F800000-0x1F8003FF - Scratch (кэш данных)
X0x1F801000-0x1F80???? - регистры аппаратуры и ќ— данные
X0x80000000-0x801FFFFF - тень RAM (кэшируема¤)
X0xa0000000-0xa01FFFFF - тень RAM (не кэшируема¤)
X0xBFC00000-0xBFC7FFFF - ROM, kernel, и shell - BIOS
X0xFFFE0130 - используетс¤ дл¤ отладки COP0.

0x00000000-0x001FFFFF - RAM
на модел¤х PSX дл¤ девелоперов ("син¤¤" и "черна¤" PSX) размер RAM увеличен в 2 и 4 раза (до 4-8 мегабайт), поэтому адреса RAM будут 0x00000000-0x003FFFFF или 0x007FFFFF. размер RAM можно узнать через 0x00000060 (см. ниже).

относительное смещение:

```
X0x00000000 - обработчик TLB miss исключени¤.
      lui    k0, 0000
      addiu       k0, k0, 0C80
      jr     (k0)
      nop
X0x00000010 - 0x0000005c - не используетс¤
X0x00000060 - размер RAM в мегабайтах. устанавливаетс¤ в SetMem().
X0x00000064 - ???
X0x00000068 - устанавливаетс¤ в 0x000000ff при RESET.
X0x0000006c - 0x0000007c - не используетс¤
X0x00000080 - обработчик основных исключений (GENERAL EXCEPTIONS).
      lui    k0, 0000
      addiu       k0, k0, 0C80
      jr     (k0)
      nop
X0x0000008c - 0x0000009c - не используетс¤
X0x000000a0 - системный вызов A0:XX
      lui    t0, 0000
      addiu       t0, t0, 05C4
      jr     (t0)
      nop
X0x000000b0 - системный вызов B0:XX
      lui    t0, 0000
      addiu       t0, t0, 05E0
      jr     (t0)
      nop
X0x000000c0 - системный вызов C0:XX
      lui    t0, 0000
      addiu       t0, t0, 0600
      jr     (t0)
      nop
X0x000000d0 - 0x000000fc - не используетс¤
X0x00000100 - указатель на таблицу обработчиков главного исключени¤.
X0x00000104 - размер таблицы в байтах.
X0x00000108 - указатель на таблицу состо¤ни¤ задачи.
X0x0000010c - размер таблицы в байтах.
X0x00000110 - указатель на таблицу потоков (TCB).
X0x00000114 - размер таблицы в байтах.
X0x00000120 - указатель на таблицу описателей событий (EVENT).
X0x00000124 - размер таблицы в байтах.
X0x00000140 - указатель на таблицу файловых дескрипторов.
X0x00000144 - размер таблицы в байтах.
X0x00000150 - указатель на таблицу драйверов устройств.
X0x00000154 - размер таблицы в байтах.
X0x0000e000 - 0x0000ffff - куча KERNEL (heap).
```

Table of Tables:

начина¤ с адреса 0x00000100 находитс¤ так называема¤ "таблица таблиц" (Table of Tables или ToT), в которой содержатс¤ указатели на все остальные важные

таблицы (событий, потоков и открытых файлов):

```
struct ToT {
	unsigned long *head;
	long size;
};

/* системна¤ ToT */
struct ToT SysToT[32];
```

всего в системной ToT может быть 32 таблицы, но используютс¤ только 6.
причем программно доступно только 4 (таблицы потоков, событий, файлов и устройств),
остальные 2 (таблица обработчиков главного исключени¤ и таблица состо¤ни¤ задачи)
используютс¤ ¤дром. только две таблицы создаютс¤ динамически (событи¤ и потоки),
остальные - строго фиксированы и заполн¤ютс¤ во врем¤ bootstrapping'a.
ниже приведены структуры всех перечисленных таблиц.

Exception Handlers - обработчики главного исключени¤:

в этой таблице наход¤тс¤ обработчики исключений процессора.

```
/* Exception Handlers Control Block */
struct ExCB {
	IntRP	*head;	/* NULL, if queue is empty */
	long flag;
};
```

при загрузке BIOS создаетс¤ четыре очереди обработчиков:
ExCB SysIntr[4];
0 - обработчик исключений от CPU (например syscall, переполнение)
1 - прерывание от счетчиков (4 обработчика)
2 - не используетс¤ (на усмотрение программиста)
3 - обработчик аппаратных прерываний

```
/* Interrupt Procedure */
struct IntRP {
	IntRP	*next;	/* NULL, if last in queue */
	int	(*func2)(int);
	int	(*func1)(void);
	long flag;	/* always 0 */
};
```

обработчик вызываетьс¤ следующим образом:
а) func2(func1()) если присутствуют обе функции, или
б) func1() если присутсвует перва¤ функци¤ но отсутствует втора¤, или
в) игнорируетс¤ в других случа¤х.

обработчики можно добавл¤ть в очередь функцией SysEnqIntRP() и убирать
функцией SysDeqIntRP().

Task Control Block - описатель потока:

таблица потоков находитс¤ в куче и у каждого процесса может быть
неограниченное количество "одновременно" выполн¤емых потоков, но по умолчанию
количество открытых потоков не должно превышать 4.
при загрузке EXE-файла с CDROM, KERNEL читает файл SYSTEM.CNF,
в котором указано максимальное количество потоков у загружаемого процесса.

```
struct TCB {
	long status;
	long mode;
	unsigned long reg[NREGS];	/* never change the offset of this */
	long system[6];		/* reserved by system */
};
```

Task State Table - таблица состо¤ни¤ задачи:

таблица состо¤ни¤ задачи хранит в себе указатель на контекст текущего потока.
эта таблица - обычный указатель на TCB, в который и записываютс¤ все регистры
при обработке исключени¤.

```
struct TCBH {
```

```
        struct TCB *entry;
        long flag;
};

TCBH TaskState;
```

функци¤ ChangeTh() мен¤ет текущий поток (путем замены указател¤ на TCB).

Event Control Block - описатель событи¤:

таблица событий находитс¤ в куче и у каждого процесса может быть
неограниченное количество обработчиков событий, но по умолчанию
количество открытых событий не должно превышать 16.
при загрузке EXE-файла с CDROM, KERNEL читает файл SYSTEM.CNF,
в котором указано количество событий у загружаемого процесса.

```
struct EvCB {
        unsigned long desc;
        long status;
        long spec;
        long mode;
        long (*FHandler)();
        long system[2];             /* reserved by system */
};
```

событие активизируетс¤ при прерывании или исключении. дл¤ этого
в системном обработчике прерываний все вхождени¤ сравниваютс¤ по
классу событи¤ (class) и вызванному прерыванию, и если они совпадают,
то управление передаетс¤ обработчику событи¤ (func).

I/O Block - дескриптор файла:

одновременно может быть открыто только 16 файлов, вне зависимости от
устройства.

```
/* io block */
struct      iob {
        int    i_flgs;
        int    i_unit;          /* pseudo device unit */
        char  *i_ma;            /* memory address of i/o buffer */
        unsigned int     i_cc; /* character count of transfer */
        unsigned long   i_offset;   /* seek offset in file */
        int    i_fstype;        /* file system type */
        int    i_errno;         /* error # return */
        struct device_table *i_dp; /* pointer into device_table */
        unsigned long    i_size;
        long    i_head;
        long    i_fd;           /* file descriptor */
};
```

Device Table - описатель устройства:

описатель устройства - фундаментальна¤ структура дл¤ работы с различными
устройствами ввода/вывода. это не об¤зательно физические устройства,
например консоль тоже входит в эту таблицу - это логическое устройство.
BIOS по умолчанию добавл¤ет в таблицу консоль (TTY), CD-ROM и карту пам¤ти.
если программе нужно использовать иные устройства (например модем, мышь или
принтер), то она должна сама добавить новый описатель.

```
/* device table */
struct device_table {
        char *dt_string;            /* device name */
        int dt_type;            /* device "type" */
        int dt_bsize;           /* file system type */
        char *dt_desc;          /* device description */
        int (*dt_init)();           /* device init routine */
        int (*dt_open)();           /* device open routine */
        int (*dt_strategy)();/* device strategy routine, returns cnt */
        int (*dt_close)();          /* device close routine */
        int (*dt_ioctl)();          /* device ioctl routine */
        int (*dt_read)();           /* fs read routine, returns count */
        int (*dt_write)();          /* fs write routine, return count */
```

```
        int (*dt_delete)();         /* file delete routine */
        int (*dt_undelete)();       /* file delete routine */
        int (*dt_firstfile)();      /* directory serach routine */
        int (*dt_nextfile)(); /* directory serach routine */
        int (*dt_format)();
        int (*dt_cd)();
        int (*dt_rename)();
        int (*dt_remove)();
        int (*dt_else)();
};
```

заполним таблицу таблиц:

```
ToT SysToT[32] = {
    SysIntr, sizeof(SysIntr),           /* 4 очереди обработчиков (по умолчанию) */
    TaskState, sizeof(TaskState),
    Tasks, sizeof(Tasks),               /* до 4 потоков (по умолчанию) */
    Events, sizeof(Events),             /* до 16 событий (по умолчанию) */
    iob_t, sizeof(iob_t),               /* 16 файлов */
    dev_t, sizeof(dev_t)                /* 10 устройств */
};
```

0x1F000000-0x1F?????? - PIO MEMORY

относительное смещение:

X0x0000 - содержит адрес процедуры main устройства (вызываетс¤ после загрузки ¤дра).
X0x0004 - должна быть строка 'Licensed by Sony Computer Entertainment Inc.\0' дл¤ проверки лицензии.
X0x0080 - содержит адрес процедуры инициализации устройства init (вызываетс¤ до загрузки ¤дра).
X0x0084 - то же, что и 0x0004
X0x00b4 - код, данные подключенного к PIO устройства.

если 0x0004 или 0x0084 не содержат строки с лицензией, то main или init не вызываютс¤.
диапазон PIO MEMORY зависит от количества пам¤ти на борту PIO-устройства.

0x1F801000-0x1F80xxxx - HARDWARE I/O
из эмул¤тора FPSE:

X0x1f801000
X0x1f801004
X0x1f801008
X0x1f80100c
X0x1f801010
X0x1f801014 - spu_delay
X0x1f801018 - dv5_delay
X0x1f80101c
X0x1f801020 - com_delay

SIO
X0x1f801040 - sio0_data
X0x1f801044 - sio0_status
X0x1f801048 - sio0_mode
X0x1f80104a - sio0_control
X0x1f80104e - sio0_baud

X0x1f801050 - sio1_data
X0x1f801054 - sio1_status
X0x1F801058 - sio1_mode
X0x1f80105a - sio1_control
X0x1f80105e - sio1_baud

X0x1f801060 - ram_size

IRQ
X0x1f801070 - int_reg

X0x1f801074 - int_mask

DMA

X0x1f801080 - mdec_dma0_madr
X0x1f801084 - mdec_dma0_bcr
X0x1f801088 - mdec_dma0_chcr

X0x1f801090 - mdec_dma1_madr
X0x1f801094 - mdec_dma1_bcr
X0x1f801098 - mdec_dma1_chcr

X0x1f8010a0 - gpu_dma_madr
X0x1f8010a4 - gpu_dma_bcr
X0x1f8010a8 - gpu_dma_chcr

X0x1f8010b0 - cd_dma_madr
X0x1f8010b4 - cd_dma_bcr
X0x1f8010b8 - cd_dma_chcr

X0x1f8010c0 - spu_dma_madr
X0x1f8010c4 - spu_dma_bcr
X0x1f8010c8 - spu_dma_chcr

X0x1f8010d0 - dma5_madr
X0x1f8010d4 - dma5_bcr
X0x1f8010d8 - dma5_chcr

X0x1f8010e0 - dma6_madr
X0x1f8010e4 - dma6_bcr
X0x1f8010e8 - dma6_chcr

X0x1f8010f0 - dma_pcr
X0x1f8010f4 - dma_icr

ROOT COUNTERS

X0x1f801100 - t0_count
X0x1f801104 - t0_mode
X0x1f801108 - t0_target

X0x1f801110 - t1_count
X0x1f801114 - t1_mode
X0x1f801118 - t1_target

X0x1f801120 - t2_count
X0x1f801124 - t2_mode
X0x1f801128 - t2_target

X0x1f801130 - t3_count
X0x1f801134 - t3_mode
X0x1f801138 - t3_target

CDROM

X0x1f801800 - cdrom_reg0
X0x1f801801 - cdrom_reg1
X0x1f801802 - cdrom_reg2
X0x1f801803 - cdrom_reg3

GPU

X0x1f801810 - gpu_reg0
X0x1f801814 - gpu_reg1

MDEC

X0x1f801820 - mdec_reg0
X0x1f801824 - mdec_reg1

SPU

(0x1f801c00-0x1f801dff)
X0x1f801d80 - spu_mvol_l
X0x1f801d82 - spu_mvol_r
X0x1f801d84 - spu_reverb_l
X0x1f801d86 - spu_reverb_r
X0x1f801d88 - spu_key_on_1
X0x1f801d8a - spu_key_on_2
X0x1f801d8c - spu_key_off_1
X0x1f801d8e - spu_key_off_2

X0x1f801d90 - spu_key_modefm_1
X0x1f801d92 - spu_key_modefm_2
X0x1f801d94 - spu_key_modenoise_2
X0x1f801d96 - spu_key_modenoise_2
X0x1f801d98 - spu_key_modereverb_1
X0x1f801d9a - spu_key_modereverb_2
X0x1f801d9c - spu_key_channelactive_1
X0x1f801d9e - spu_key_channelactive_2

X0x1f801da6 - spu_sbaddr
X0x1f801da8 - spu_data
X0x1f801daa - spu_reg0
X0x1f801dac - spu_reg1
X0x1f801dae - spu_status

X0x1f801db0 - spu_cdvol_l
X0x1f801db2 - spu_cdvol_r
X0x1f801db4 - spu_extvol_l
X0x1f801db6 - spu_extvol_r

X0x1f801dc0 - spu_reverbconfig
X0x1f801dfc - spu_factor_l
X0x1f801dfe - spu_factor_r

DEBUG
X0x1f802030 - int_2000
X0x1f802040 - dip_switches

0xBFC00000-0xBFC7FFFF - BIOS

относительное смещение:

X0x00000100 - timestamp, или дата компил¤ции BIOS (например 0x19951204 - 4 декабр¤ 1995)
X0x00000104 - строка 'Sony Computer Entertainment Inc.'
X0x0000012c - модель PSX, использующа¤ этот BIOS. например 'CEX-3000/1001/1002 by K.S.'
X0x00000150-0x0000ffff - Initialisation of lots of stuff, some system calls,...
X0x00010000-0x00017fff - This contains most of the kernel system calls code.
It is copied to RAM at 0xa0000500. After it is copied, the kernel jumps
to 0xa0000500, where it clears some memory, (used internally by kernel operations).
X0x00018000-0x007fffff - This where the Shell is located (CD Player memory card
management,etc...). It is copied to main memory at address 0x80030000, and its
there that is decided what to do, do the country check, and stuff, and if it
should return to kernel to execute an exe file.

---

si17911@ci.uminho.pt
kvzorganic@mail.ru

назад...