BIOS

BIOS от PSX представляет собой ROM-микросхему, объемом 512KB (в 8 раз больше чем у PC !!!). внутри него зашиты ядро OS (KERNEL) и графическая оболочка (проигрыватель музыкальных CD и менеджер карт памяти). существует несколько версий BIOS для разных моделей PSX и разных регионов (стран), я выбрал SCPH1001.BIN.

СИСТЕМНЫЕ ВЫЗОВЫ KERNEL

KERNEL находится внутри BIOS. после сброса PSX он грузится BIOS'ом в оперативку. KERNEL содержит API, в состав которого входят все основные функции для работы с PSX на уровне приложений. вызов функций ядра производится следующим образом: в регистр **t1** помещается номер функции. потом происходит вызов на шлюз. всего у ядра три шлюза - **A0**, **B0** и **C0**. допустим нам нужно вызвать функцию ядра **A0:3F**. мы делаем так:

```
addiu t1, zr, 003F  ; поместить номер функции в регистр t1
jal   000000A0      ; перейти на функцию
nop
```

или вот так:

```
addiu t2, zr, 00A0
jr    t2            ; перейти на функцию
addiu t1, zr, 003F  ; поместить номер функции в регистр t1
```

на данный момент практически все названия функций уже выяснены. например предыдущая функция - это **printf**.
если у функции есть параметры, то они передаются через регистры **a0**, **a1**, **a2** и **a3** и далее (если параметров больше четырех), через стек **sp+0010**. выходное значение функции (если есть) содержится в регистре **v0**.
в этом документе весь список функций ядра. а в этом немного об устройстве памяти PSX.

ИНСТРУКЦИЯ SYSCALL

у PSX также есть аппаратная поддержка системных вызовов, но она не используется так широко, как API. это инструкция syscall. всего существует четыре таких вызова:

```
// вызвать GENERAL EXCEPTION
void Exception()
{
  _asm
  {
    addiu  a0, zr, 0
    syscall
    jr     ra
    nop
  }
}

// войти в критическую секцию (запретить прерывания)
void EnterCriticalSection()
{
  _asm
  {
    addiu  a0, zr, 1
    syscall
    jr     ra
    nop
  }
}

// выйти из критической секции (разрешить прерывания)
void ExitCriticalSection()
{
  _asm
  {
    addiu  a0, zr, 2
    syscall
    jr     ra
    nop
  }
}

// переключить задачу
void SwitchTask(TCB *tcb)
{
  _asm
```

```
  {
    addiu   a0, zr, 3
    addiu   a1, zr, tcb
    syscall
    jr      ra
    nop
  }
}
```

здесь в качестве аргумента используется не поле параметра инструкции, а регистр **a0**. эти функции используются в системных программах: обработчики событий и прерываний, функции работы с железом итп.

назад...