

Pcsx 1.5 with debugger Ver7

Первая секция:

F11 – запуск дебаггера.



Выполнить инструкцию:

Step, средняя кнопка мыши.

Продолжить эмуляцию:

Run, **ESC**, двойной клик левой кнопки мыши.

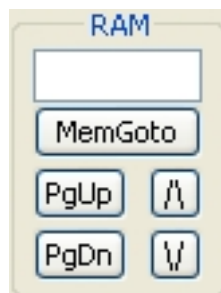


Точки останова для: чтения/записи по **\$** адресу памяти, **=** значения, **PC** выполняемой инструкции, **Reg** регистров.

Memory Write/Read – **\$** минимальный адрес <-> максимальный адрес = значение.

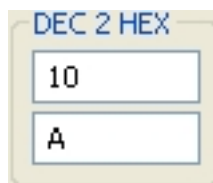
* Точки останова: минимальный **\$** адрес и **=** значение, могут использоваться независимо.

*Доступен выбор операций сравнения: **=**, **!**, **>**, **<**.

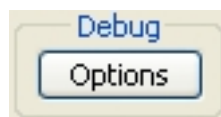


Управление окном памяти.

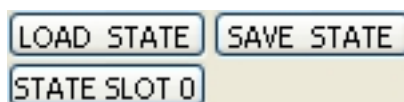
Вторая секция:



Конвертер из десятичной системы в шестнадцатеричную и обратно.



Установка параметров отладчика.



Загрузка/сохранение игры.

Load RAM	Open Dump
Dump RAM	

Загрузить/сдампить память: “\dump\GAME_ID_RAM.bin” , “ \dump\GAME_ID_VRAM.bin” .

Открыть папку дампа.

*Дамп использует сейвстейт для корректной работы.(этот параметр можно изменить в Debug->Options)

Третья секция:

Save	Clear	Load

Окно регистрации точек останова.

Загрузить/сохранить лог: “\logs\GAME_ID.txt”.

*Также окно может использоваться для документирования различной информации.

<input type="checkbox"/> CD-ROM Read	<input type="checkbox"/> On sector	0	< - >	0
--------------------------------------	------------------------------------	---	-------	---

Точка останова для чтения данных с компакт диска.

Точка останова для чтения указанного сектора или диапазона секторов, компакт диска.

* “On sector” работает только совместно с “CD-ROM Read”.

* LBA секторов указываются в шестнадцатеричной системе.

Registers Reg at at <div style="border: 1px solid black; height: 20px; width: 100%;"></div> <div style="border: 1px solid black; padding: 2px; width: 100%;">Patch</div>	Mem Patch Address <div style="border: 1px solid black; height: 20px; width: 100%;"></div> <div style="display: flex; align-items: center;"> <div style="margin-right: 10px;"> <input checked="" type="radio"/> 1 Byte <input type="radio"/> 2 Bytes <input type="radio"/> 4 Bytes </div> <div style="border: 1px solid black; height: 20px; width: 100%;"></div> </div> <div style="display: flex; justify-content: flex-end; gap: 10px;"> <div style="border: 1px solid black; padding: 2px;">Patch</div> <div style="border: 1px solid black; padding: 2px;">Get</div> </div>
---	--

Патчеры регистров и памяти.

Размер пропатчиваемых данных вычисляется автоматически.(этот параметр можно изменить в Debug->Options)

SPU WriteRegister		
<input type="checkbox"/> Volume	<input type="checkbox"/> Frequency	<input type="checkbox"/> Start address
<input type="checkbox"/> SPU levels1	<input type="checkbox"/> SPU levels2	<input type="checkbox"/> SPU levels 3
<input type="checkbox"/> spu-mem adr	<input type="checkbox"/> data2spu	<input type="checkbox"/> SPU control
<input type="checkbox"/> SPU status	<input type="checkbox"/> SPU irq	<input type="checkbox"/> SPU on 0-16
<input type="checkbox"/> SPU on 16-24	<input type="checkbox"/> SPU off 0-16	<input type="checkbox"/> SPU off 16-24

Точка останова SPU.

GPU	
<input type="checkbox"/> Data log	<input type="checkbox"/> Control log

Точка останова GPU.

DMA

☐ CPU2SPU DMA

☐ SPU2CPU DMA

Точка останова DMA.

Управление окном инструкций.

Дополнительные возможности:

Получение адреса из окна памяти, выделите строку и кликните по столбцу.

Переход в режим ASCII – двойным кликом левой кнопкой мыши в окне памяти.

Получение данных из окон регистров и инструкций - двойным кликом левой кнопкой мыши.

Использование значения регистров по его имени. *Используйте @0 - @3 вместо a0 - a3 регистров.

MEMORY																	
Address	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	Nº
00011B50	fd	ff	80	1c	04	00	63	24	0c	00	10	26	0b	00	03	3c	00
00011B60	07	00	02	24	08	89	62	ac	21	20	40	00	80	18	10	00	01
00011B70	00	00	60	ac	04	00	63	24	00	00	60	ac	04	00	63	24	02
00011B80	00	00	60	ac	04	00	63	24	00	00	60	ac	fc	ff	84	24	03
00011B90	04	00	82	28	f6	ff	40	10	04	00	63	24	05	00	80	18	04
00011BA0	00	00	00	00	00	00	60	ac	ff	ff	84	24	fd	ff	80	1c	05
00011BB0	04	00	63	24	f4	ff	10	26	04	00	11	24	13	00	12	24	06
00011BC0	18	00	f0	ae	21	10	00	02	0b	00	03	3c	21	f0	00	02	07
00011BD0	08	89	64	ac	35	51	00	0c	1c	00	e0	ae	21	80	c0	03	08
00011BE0	20	00	a5	8f	18	00	b2	8f	14	00	b1	8f	1c	00	e5	ae	09
00011BF0	1c	00	a5	8f	04	00	39	37	18	00	e5	ae	10	00	a5	8f	0A
00011C00	0b	00	02	3c	02	00	00	12	08	89	45	ac	04	00	39	3b	0B
00011C10	2c	00	bf	8f	28	00	be	8f	08	00	e0	03	30	00	bd	27	0C
00011C20	ff	f0	18	3c	80	10	10	00	20	00	42	8c	ff	ff	18	37	0D
00011C30	24	c0	58	00	25	c0	b8	02	80	10	10	00	08	00	e0	03	0E
00011C40	20	00	58	ac	e8	ff	bd	27	80	10	10	00	14	00	bf	af	0F
00011C50	10	00	be	af	18	00	55	8c	00	00	00	00	0b	00	a0	12	10
00011C60	21	f0	00	02	21	80	a0	02	08	47	00	0c	00	09	15	3c	11
00011C70	80	10	10	00	00	00	58	8c	01	00	02	3c	03	00	00	13	12
00011C80	a0	1c	55	24	d4	32	02	0c	00	00	00	00	21	80	c0	03	13
00011C90	14	00	bf	8f	10	00	be	8f	08	00	e0	03	18	00	bd	27	14
00011CA0	e8	ff	bd	27	10	00	bf	af	08	47	00	0c	00	0a	15	3c	15
00011CB0	80	10	10	00	00	00	58	8c	01	00	02	3c	03	00	00	13	16
00011CC0	a0	1c	55	24	d4	32	02	0c	00	00	00	00	10	00	bf	8f	17

Использование координат памяти в качестве адреса:

В любое поле, введите: идентификатор памяти(m), номер столбца(y) и номер строки(x).

Пример: m47 =11bc4.

Mem Patch

Address

☒ 1 Byte
 ☐ 2 Bytes
 ☐ 4 Bytes

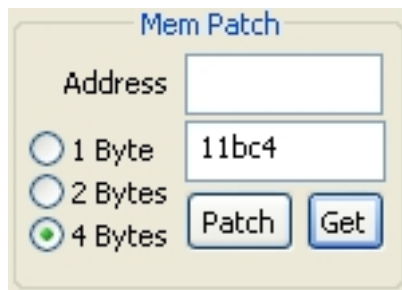
Mem Patch

Address

☒ 1 Byte
 ☐ 2 Bytes
 ☐ 4 Bytes

Получение значений из регистров и координат памяти:

Введите значение в поле данных, и нажмите кнопку Get.



Чтение данных из памяти:

Введите адрес памяти и в зависимости от нужного количества байт, нажмите на кнопку Byte.

*Данная функция работает, только если активирован флаг Debug->Options->Auto calculation of patching data size.

Псевдо инструкции:

*Синтаксис псевдо инструкций взят из **PSIG** (PlayStation Instructions Generator program).

- **LI** v0, 8550 - загрузить константу.
- **LZR** v0 - загрузить ноль.
- **LONE** v0 - загрузить единицу.
- **LMAX** v0 - загрузить максимальное значение(0xFFFFFFFF).
- **INC** v0 - увеличить на единицу.
- **INCW** v0 - увеличить на единицу без знаковое число.
- **DEC** v0 - уменьшить на единицу.
- **DECU** v0 - уменьшить на единицу без знаковое число.
- **COPY** v0, v1 - скопировать значение второго регистра в первый.
- **GETB** v1, v0 - получить младший байт(из второго регистра в первый).
- **GETW** v1, v0 - получить младшее слово (из второго регистра в первый).
- **NEG** v0, v1 - сделать число отрицательным.
- **NEGU** v0, v1 - сделать без знаковое число отрицательным.
- **NOT** v0, v1 - инверсия всех бит в v1.

- **BNEZ** a0, label - прыгаем если a0 не равно нулю.
- **BEQZ** a1, label - прыгаем если a1 равно нулю.
- **SKIP** label - безусловный прыжок с относительным адресом.
- **SKIR** label - безусловный прыжок с относительным адресом.(перед прыжком сохраняется адрес возврата в регистр ra)

- **SZB** aac8(v0) - сохранить байт равный нулю.
- **SZH** 0000(v1) - сохранить полуслово равное нулю.
- **SZW** 1c(v0) - сохранить слово равное нулю.

- **SPU** - шаг на вершину стека.
- **SPD** - шаг на дно стека.
- **SPSU** 8 - перемещение вверх по стеку.
- **SPSD** 8 - перемещение вниз по стеку.

- **SREG** v0, 38 - сохранить регистр в стек.
- **LREG** v1, 38 - загрузить регистр из стека.
- **SRET** 1c - сохранить адрес возврата в стек.
- **LRET** 1c - загрузить адрес возврата из стека.
- **RET** - выйти из функции.

- **NOOP** - инструкция не выполняющая никаких действий.

Реализация дебаггера: **zHAOSiLi[EGCG]**

Дополнительные возможности: **Zidane, HoRRoR, Mr2**