

Pcsx 1.5 with debugger Ver7

First section:

F11 – start debugger.

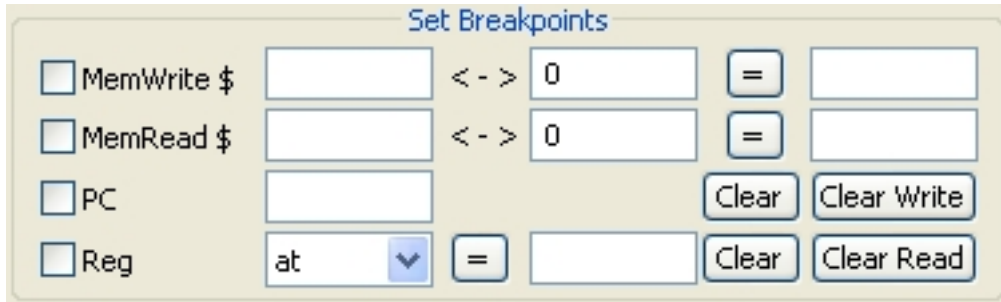


Execute instruction:

Step, middle mouse button.

Continue emulation:

Run, **ESC**, left mouse button double click.

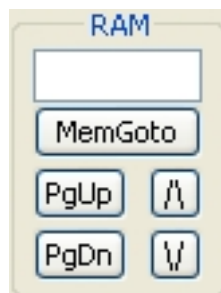


Breakpoints for: memory **write/read** \$ address, = value, **PC** executable instruction, **Register**.

Memory Write/Read – \$ min address <-> max address = value.

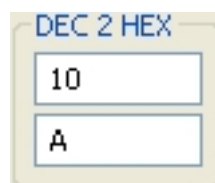
* Breakpoints: min \$ address and = value, can be used independently.

* Available comparison operations: =, !, >, <.

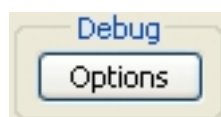


Memory window control buttons.

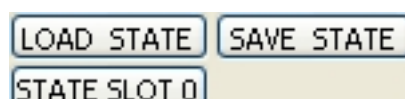
Second section:



Dec to hex to dec value converter.



Show debugger options window.



Load/save game buttons.

Load RAM

Open Dump

Dump RAM

Load/dump: “\dump\GAME_ID_RAM.bin” , “ \dump\GAME_ID_VRAM.bin”.
 Open dump folder.

*Dump uses save state for the correct work.(This parameter can be changed in Debug->Options)

Third section:

Save

Clear

Load

The breakpoints log window.

Load/save log: “\logs\GAME_ID.txt”.

*The window can also be used to log of user information.

☐ CD-ROM Read
 ☐ On sector

0 < - > 0

CD-ROM read breakpoint.

Sector read or sectors range breakpoint.

* "On sector" works only with "CD-ROM Read".

* LBA sectors are set in the hexadecimal system.

Registers

Reg at

Patch

Mem Patch

Address

☒ 1 Byte

☐ 2 Bytes

☐ 4 Bytes

Patch

Get

Registers/memory patchers.

Auto – auto calculation of patching data size.

SPU WriteRegister

☐ Volume

☐ Frequency

☐ Start address

☐ SPU levels1

☐ SPU levels2

☐ SPU levels 3

☐ spu-mem adr

☐ data2spu

☐ SPU control

☐ SPU status

☐ SPU irq

☐ SPU on 0-16

☐ SPU on 16-24

☐ SPU off 0-16

☐ SPU off 16-24

SPU breakpoints.

GPU

☐ Data log

☐ Control log

GPU breakpoints.

DMA

☐ CPU2SPU DMA
☐ SPU2CPU DMA

DMA breakpoints.

PgUp
PgDn
↖
↘
PC
Show
Jump
93e34

PC window control buttons.

Other functions:

To get the address from the memory window, select the line and click on the column number.
 For switching to ASCII mode, make a double click with left mouse button, in the memory window.
 To get a data from registers and instructions windows, make a double click with left mouse button.
 Use the register name as its value. *Use the @0 - @3 to setup a0 - a3 registers.

| 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 |

Using memory coordinate as address:

Enter in any field: the memory ID (M), column number (Y) and line number (x).

Example: m47 =11bc4.

Mem Patch

Address

☒ 1 Byte
 ☐ 2 Bytes
 ☐ 4 Bytes

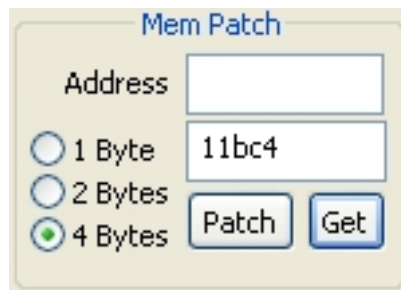
Mem Patch

Address

☒ 1 Byte
 ☐ 2 Bytes
 ☐ 4 Bytes

Getting a values from registers and memory coordinates:

Enter a value in data field, and press "Get" button.



Reading data from memory:

Enter the memory address, and depending on the required number of bytes, click on the “Byte” button.

* This function works only if the “Debug->Options->Auto calculation of patching data size” flag is activated.

Pseudo instructions:

*Pseudo instructions are taken from **PSIG** (PlayStation Instructions Generator program).

- **LI** v0, 8550 - load immediate.
- **LZR** v0 - load zero to register.
- **LONE** v0 - load one to register.
- **LMAX** v0 - load maximum value(0xFFFFFFFF).
- **INC** v0 - increment.
- **INCUI** v0 - increment unsigned.
- **DEC** v0 - decrement.
- **DECU** v0 - decrement unsigned.
- **COPY** v0, v1 - copy second register to first.
- **GETB** v1, v0 - get low byte(from second register to first).
- **GETW** v1, v0 - get low word(from second register to first).
- **NEG** v0, v1 - make a number negative.
- **NEGU** v0, v1 - make a unsigned number negative.
- **NOT** v0, v1 - inversion of all bits in v1.

- **BNEZ** a0, lable - jump if a0 is not equal zero.
- **BEQZ** a1, lable - jump if a1 is equal zero.
- **SKIP** label - unconditional jump with PC-relative address.
- **SKIR** label - unconditional jump with PC-relative address. (before jumping, saves the return address in the ra register)

- **SZB** aac8(v0) - store byte equal zero.
- **SZH** 0000(v1) - store halfword equal zero.
- **SZW** 1c(v0) - store word equal zero.

- **SPU** - stack pointer, one step up on top.
- **SPD** - stack pointer, one step down to bottom.
- **SPSU** 8 - stack pointer, some steps up on top.
- **SPSD** 8 - stack pointer, some steps down to bottom.

- **SREG** v0, 38 - save register to stack.
- **LREG** v1, 38 - load register from stack.
- **SRET** 1c - save return address to stack.
- **LRET** 1c - load return address from stack.
- **RET** - return from function.

- **NOOP** - instruction takes no action.