

Code marker

Mark at start

1. Если нужно отследить все функции задействованные в игре.
 2. Если нужно отследить код с текущего момента, или загрузки сохранения.
- * Если известна точка завершения разметки, можно установить точку остановки.

Mark from

1. Если нужно отследить код с указанного момента до завершения в указанной точке.

Patch

1. Если нужно вручную изменить ход выполнения кода.

Маркированные данные:

1_call_marked.bin – размеченные вызовы функций.

1_code_marked.bin – размеченный отработанный код и используемые переменные.

Данные разметки хранятся в папке “dump”.

dump\1_call_marked.bin

dump\1_code_marked.bin

Коды разметки:

Call:

"2" - Функция.

Code:

"1" - прочитанный байт.

"2" - записанный байт.

"3" - прочитанный и записанный байт.

"7" - выполненная инструкция.

Консольные сообщения маркировщика:

Mark at start:

Code Marker: marking start – разметка началась.

Code Marker: marking stoped – разметка остановлена.

Code Marker: marking restarted – разметка продолжается, после каких-либо манипуляций.

Mark from:

Code Marker: marking enabled – разметка начнётся с указанной точки.(PC = start)

Code Marker: marking disabled – разметка отключена.

Code Marker: marking reset – начальная точка разметки переустановлена.

*Включить отладочную консоль:

Configuration->CPU->Enable Console Output

Анализ:

Путём сравнения дампов, ищутся задействованные функции\переменные.

Затем с помощью дебаггера и IDA(The Interactive Disassembler) ищется конкретная функция\переменная.

Пример:

Игра: Tomb Rider III Adventures of Lara Croft.

Game ID: SLUS-00691

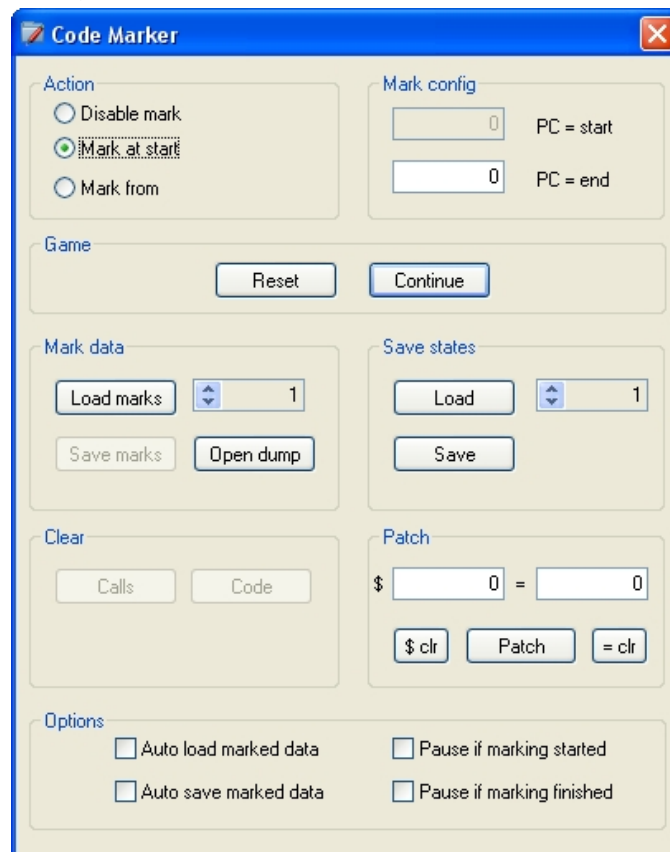
Цель: найти код открывания двери рычагом в первом уровне.



1. Через окно маркировщика, загружаем сохранение неподалёку от рычага.(кнопка Load)



2. Включаем разметку.(Mark at start)



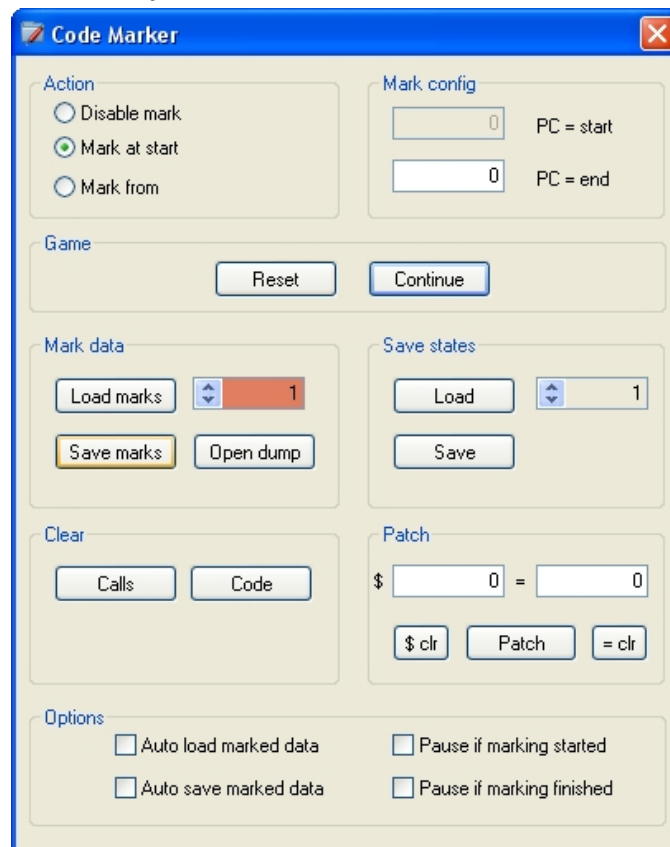
3. Жмём кнопку Continue.

4. Жмём кнопку X.

5. Подходим к рычагу вплотную.(R1+UP)



6. Сохраняем данные разметки под **номером 1.**(Save marks)

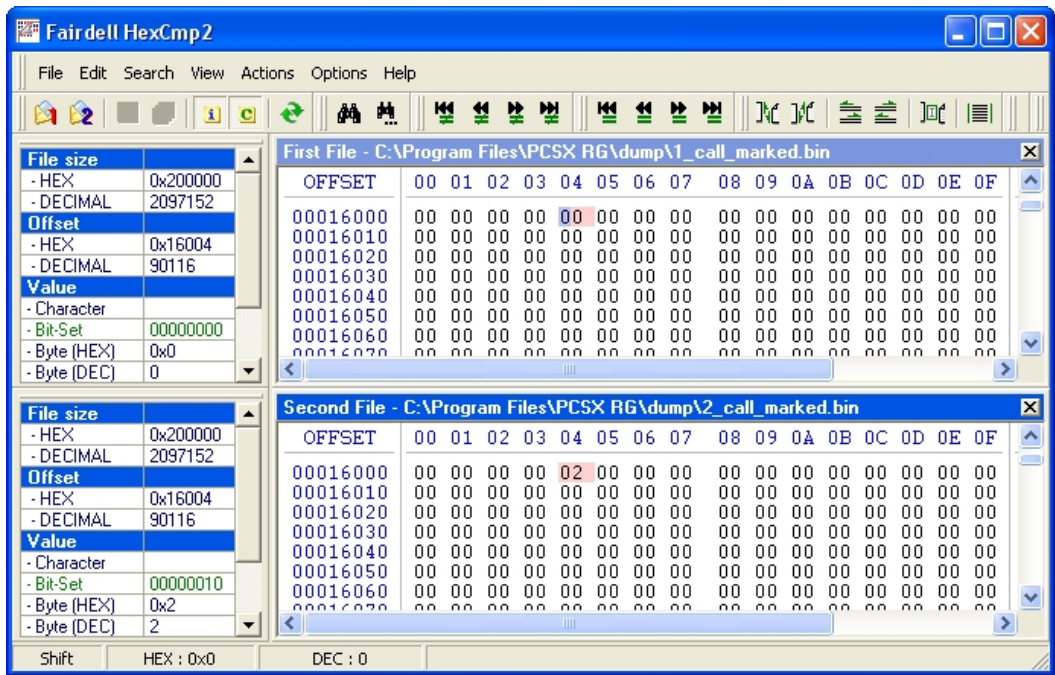


7. Дёргаем рычаг, ждём пока откроется дверь.



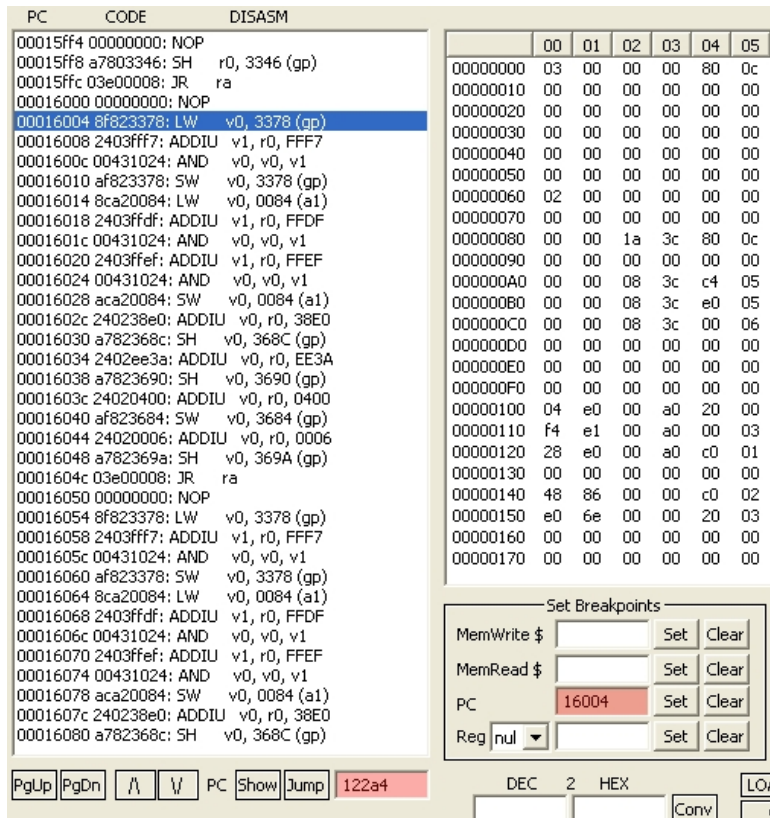
8. Сохраняем данные разметки под **номером 2.**(Save marks)

9. Путём сравнение меток, получаем список используемых функций для открывания двери.



На скриншоте мы видим метку “2” по адресу \$16004, это означает что тут начинается функция связанная с открыванием двери.

10. Для того чтобы узнать кто вызывает эту функцию нужно использовать дебаггер.



Устанавливаем брэйкпоинт PC=16004, дёргаем рычаг, и получаем код вызова функции \$122a4.

Чтобы узнать, имеет ли функция f-> 16004, непосредственное отношение к открыванию двери, нам нужно отключить её вызов:

PC	CODE	DISASM
00012294	3c010009: LUI	at, 0009
00012298	00220821: ADDU	at, at, v0
0001229c	8c22ceac: LW	v0, CEAC (at)
000122a0	00000000: NOP	
000122a4	0040f809: JALR	v0, ra
000122a8	02202821: ADDU	a1, s1, r0
000122ac	96030050: LHU	v1, 0050 (s0)
000122b0	00000000: NOP	
000122b4	246400b6: ADDIU	a0, v1, 00b6
000122b8	3082ffff: ANDI	v0, a0, FFFF
000122bc	2c42016d: SLTIU	v0, v0, 016D
000122c0	10400003: BEQ	v0, r0, 000122D0
000122c4	00031400: SLL	v0, v1, 10
000122c8	080048bb: J	000122EC
000122cc	a6000050: SH	r0, 0050 (s0)
000122d0	00021403: SRA	v0, v0, 10
000122d4	2842ff4a: SLTI	v0, v0, FF4A
000122d8	10400003: BEQ	v0, r0, 000122E8
000122dc	2462ff4a: ADDIU	v0, v1, FF4A
000122e0	080048bb: J	000122EC
000122e4	a6040050: SH	a0, 0050 (s0)
000122e8	a6020050: SH	v0, 0050 (s0)
000122ec	978333d8: LHU	v1, 33D8 (gp)
000122f0	00000000: NOP	
000122f4	2464016c: ADDIU	a0, v1, 016C
000122f8	3082ffff: ANDI	v0, a0, FFFF
000122fc	2c4202d9: SLTIU	v0, v0, 02D9
00012300	10400004: BEQ	v0, r0, 00012314
00012304	00031400: SLL	v0, v1, 10
00012308	a78033d8: SH	r0, 33D8 (gp)
0001230c	080048cd: J	00012334
00012310	00000000: NOP	
00012314	00021403: SRA	v0, v0, 10
00012318	2842fe94: SLTI	v0, v0, FE94
0001231c	10400004: BEQ	v0, r0, 00012330
00012320	2462fe94: ADDIU	v0, v1, FE94

PgUp PgDn PC Show Jump

Start address: PU levels 3, PU control, PU on 0-16, PU off 16-24

Reg Patch: Reg: [dropdown], Reg. Patch

Mem Patch: Address: 122a4, [1 Byte], [2 Bytes], [4 Bytes] Patch

Загружаем сохранение возле рычага.

В поле Address впишите: 122a4.

В поле Bytes впишите: 0.

Переключатель установите в положение: 4 Bytes.

Нажмите кнопку Patch.

Снова дёргаем рычаг, дверь по-прежнему открывается, но при этом Лара перестала ходить. А это не то что нам нужно, значит переходим к следующей метке, и повторяем анализ заново.

11. Поочерёдно отключаем каждую функцию, и проверяем откроется ли дверь. Если дверь не открылась, значит мы нашли нужную нам функцию. Далее исследуем её через IDA.

```

TEXT:0001C9DC
TEXT:0001C9DC loc_1C9DC: # DATA XREF: TEXT:000948F8↓o
* TEXT:0001C9DC sll $v0, $s2, 4
* TEXT:0001C9E0 addu $v0, $s2
* TEXT:0001C9E4 lw $v1, 0x34A4($gp)
* TEXT:0001C9E8 sll $v0, 3
* TEXT:0001C9EC li $a3, 6
* TEXT:0001C9F0 beq $s4, $a3, loc_1CA04
* TEXT:0001C9F4 addu $s0, $v1, $v0
* TEXT:0001C9F8 li $v0, 9
* TEXT:0001C9FC bne $s4, $v0, loc_1CA18
* TEXT:0001CA00 nop
TEXT:0001CA04
TEXT:0001CA04 loc_1CA04: # CODE XREF: sub_1C5BC+434↑j
* TEXT:0001CA04 lhu $v0, 0x28($s0)
* TEXT:0001CA08 nop
* TEXT:0001CA0C andi $v0, 0x80
* TEXT:0001CA10 bnez $v0, loc_1D004
* TEXT:0001CA14 andi $v0, $s6, 0x8000
TEXT:0001CA18
TEXT:0001CA18 loc_1CA18: # CODE XREF: sub_1C5BC+440↑j
* TEXT:0001CA18 li $a3, 2
* TEXT:0001CA1C bne $s4, $a3, loc_1CA30
* TEXT:0001CA20 nop
* TEXT:0001CA24 lhu $v0, 0x28($s0)
* TEXT:0001CA28 j loc_1CA50
* TEXT:0001CA2C andi $v0, 0x40
TEXT:0001CA30 # -----
TEXT:0001CA30
TEXT:0001CA30 loc_1CA30: # CODE XREF: sub_1C5BC+460↑j

```

Рекомендации:

Для продуктивного анализа кода, маркировщик нужно использовать совместно с дебаггером “debugger Ver2”.

Ссылка: <http://www.romhacking.net/utilities/267/>

*Можно скопировать исполняемый файл “pcsx.exe” в папку маркировщика.