Цель и задачи

Изучить процесс загрузки одноплатного компьютера HiFive Unmatched через реверс-инжиниринг стартового кода с использованием open-source инструментов.

1. Провести дизассемблирование предоставленного дампа ROM, содержащего ZSBL.
2. Определить назначение отдельных блоков кода.
3. Реализовать отображение результатов исполнения инструкций в Radare2.
4. Систематизировать результаты анализа.

Дизассемблирование ZSBL

Отображение результата выполнения инструкций в R2

Заключение

Дизассемблирование ZSBL. Выбор инструментов

objdump

+	-
Фундаментальный анализ бинарного файла	Не предоставляет инструментов для анализа кода

Iaito (GUI for Radare2)

+	-
Все преимущества Radare2	GUI (вылет при попытке вывести в консоль/файл более 2550 строчек дизассемблированного кода)
Встроенная система контроля версий	
GUI (удобное управление и т.д.)	

Radare2

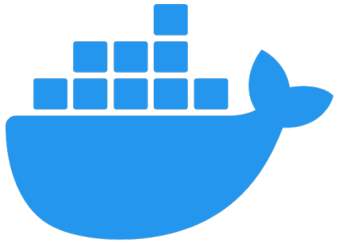
+	-
Неплохой автоматический анализ	Отсутствие GUI
Возможность вывода описаний инструкций в комментарии	Некорректное распознавание сжатых инструкций

Ghidra

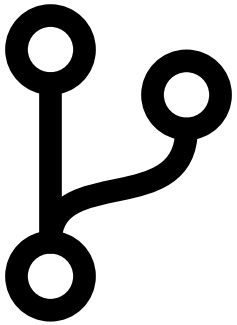
+	-
Хороший автоматический анализ	Проприетарный формат хранения данных проекта
Возможность совместной работы	Отсутствие удобных инструментов экспорта результатов
Удобные инструменты для разметки кода	

Дизассемблирование ZSBL. Сервер Ghidra

Ghidra Server



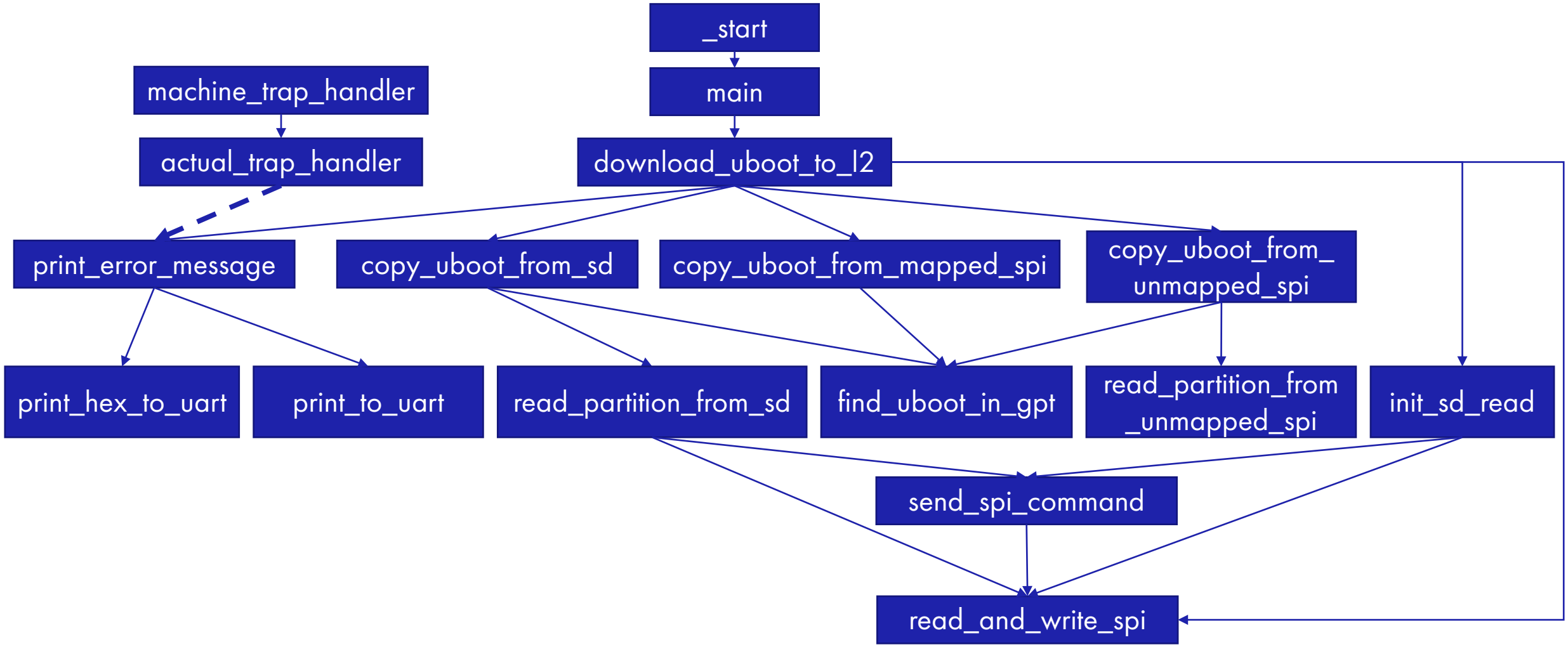
docker



СКВ

```
PORTS                                                                    NAMES
0.0.0.0:13100-13102→13100-13102/tcp, :::13100-13102→13100-13102/tcp      ghidra-server
```

Дизассемблирование ZSBL. Call Flow Graph.



Дизассемблирование ZSBL. Карта ROM (1/2)

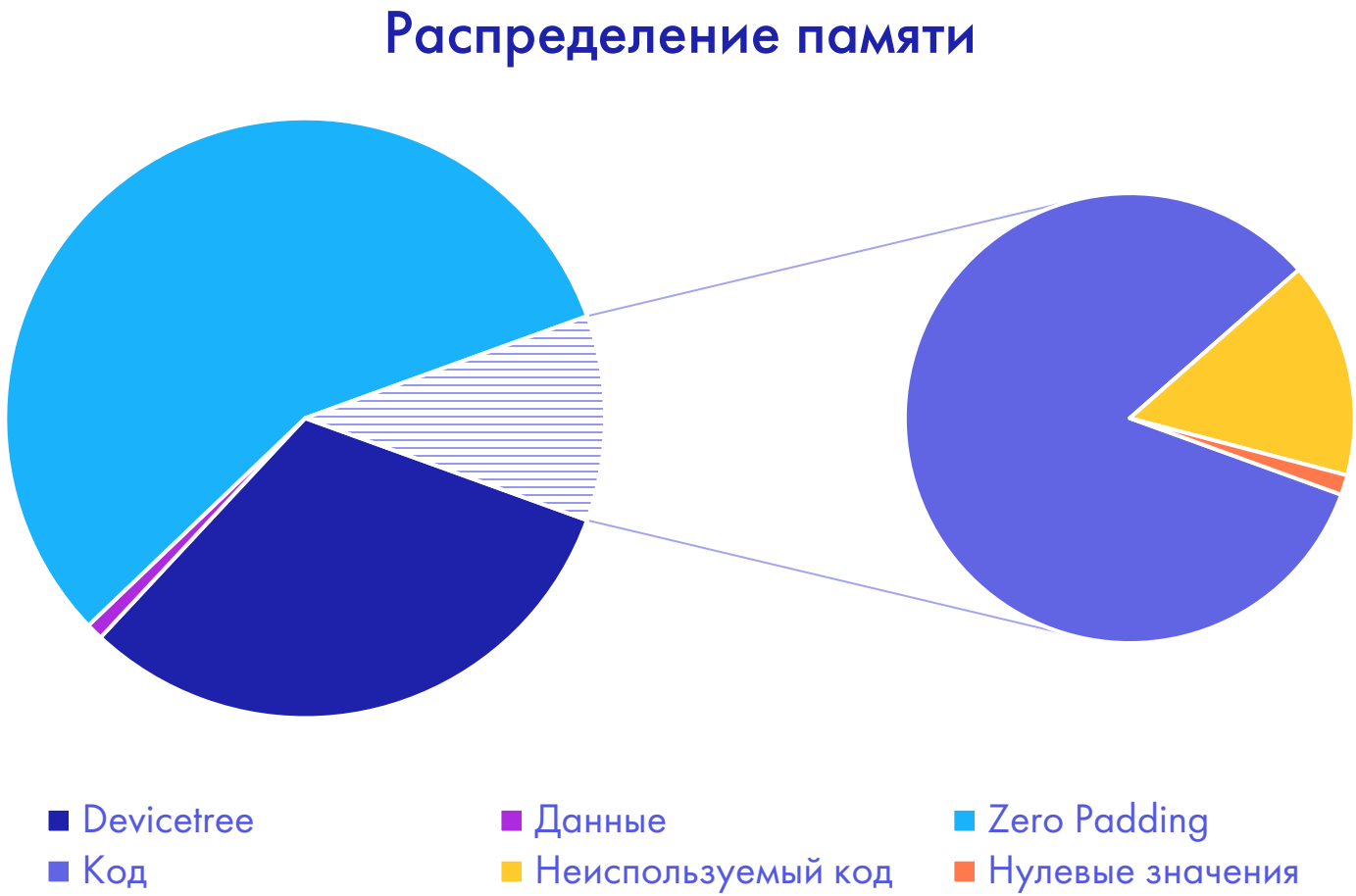
Адрес	Содержание
0x10000-0x10107	Инициализация
0x10108-0x1010b	Трамплин в Machine Trap Handler
0x10140-0x10149	Machine Trap Handler
0x10176-0x10225	Main()
0x10258-0x10275	Чтение и запись в SPI
0x10276-0x1032b	Чтение тома (Flash Bit-banged)
0x10348-0x10363	Отправка сообщения по UART
0x10364-0x1038f	Отправка шестнадцатеричного числа по UART
0x103ec-0x104bf	Поиск и копирование кода U-Boot (Flash Bit-banged)
0x104c0-0x10599	Поиск и копирование кода U-Boot (SD Bit-Banged)
0x1059a-0x10637	Поиск и копирование кода U-Boot (Flash Memory-Mapped)

Дизассемблирование ZSBL. Карта ROM (2/2)

Адрес	Содержание
0x10686-0x10703	Вывод кода ошибки по UART
0x10704-0x10953	Загрузка U-Boot в L2 кэш
0x10980-0x109d5	Поиск U-Boot в считанной GPT
0x10aba-0x10b53	Отправка команды по SPI
0x10b54-0x10d0d	Инициализация чтения с SD карты
0x10d0e-0x10e27	Чтение тома (SD Card Bit-Banged)
0x10e80-0x136b5	Devicetree
0x136b8-0x1372f	LUTs
0x13748-0x13753	Строка `Error 0x`
0x13768-0x13777	GUID искомого тома
0x13788-0x17fff	Zero Padding

Дизассемблирование ZSBL. Анализ кода

- ZSBL написан изначально на Assembler
- Стиль написания кода неоднороден
- Присутствуют фрагменты неиспользованного и недостижимого кода
- Условия нескольких операторов перехода постоянны



Дизассемблирование ZSBL. Анализ кода

Ожидание окончания операции PDMA

```
...
        wait_for_end_copying
0001061c    c.lw      a5,0x0 (a4=>DAT_03080000)
0001061e    c.andi    a5,0x2
00010620    c.bnez    a5,wait_for_end_copying
00010622    c.lw      a5,0x0 (a4=>DAT_03080000)
00010624    c.li      a3,0x1
00010626    srliw     a5,a5,0x1e
0001062a    c.addiw   a5,0x0
0001062c    bne       a5,a3,return_0
00010630    sb        zero,0x0 (a4=>DAT_03080000)
00010634    c.li      a0,0x0
00010636    c.j       return
...
```

Неиспользуемый фрагмент кода

```
...
        unused_wait_for_pdma
00010666    c.lw      a5,0x0 (a4=>DAT_03080000)
00010668    c.andi    a5,0x2
0001066a    c.bnez    a5,unused_wait_for_pdma
0001066c    c.lw      a5,0x0 (a4=>DAT_03080000)
0001066e    c.li      a3,0x1
00010670    srliw     a5,a5,0x1e
00010674    c.addiw   a5,0x0
00010676    bne       a5,a3,unused_return_error_0xd
0001067a    sb        zero ,0x0 (a4=>DAT_03080000)
0001067e    c.li      a0,0x0
00010680    ret
...
```

Дизассемблирование ZSBL. Анализ кода

Копирование кода вместо использования функции read_and_write_spi

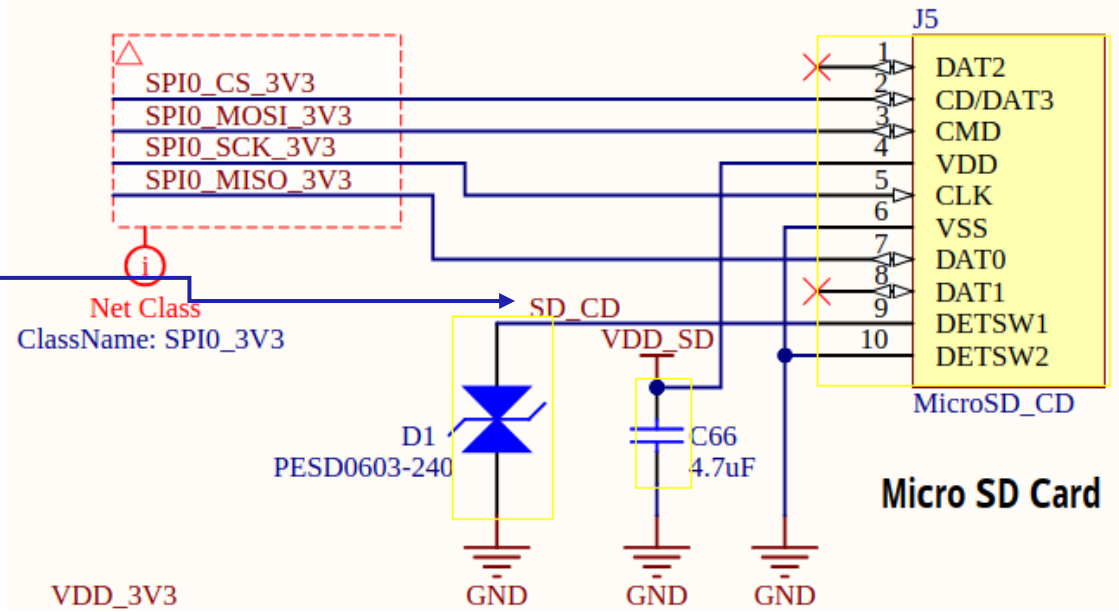
```
; function a5 = read_and_write_spi (qspi_addr, msg_to_send)
                        read_and_write_spi
00010258  addi     a5,qspi_addr,0x48              ; do{
0001025c  amoor.w  a5,msg_to_send,(a5)            ; qspi_txdata |= msg_to_send
00010260  slli     a4,a5,0x20
00010264  blt      a4,zero,read_and_write_spi     ; } while(QSPI_TX_FIFO_FULL)
                        reading_loop
00010268  c.lw     a5,0x4c(qspi_addr)             ; do {
0001026a  c.addiw  a5,0x0
0001026c  blt      a5,zero,reading_loop           ; } while(QSPI_RX_FIFO_EMPTY)
00010270  andi     qspi_addr,a5,0xff              ; return qspi_rxdata[7:0]
00010274  ret
```

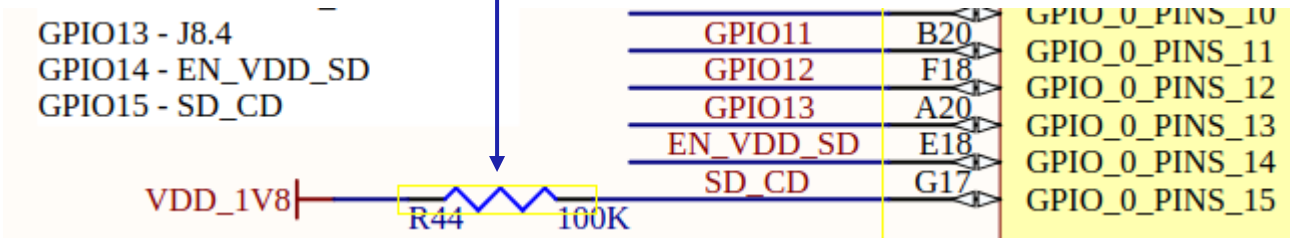
```
...
0001029a  srliw    a4,address,0x10                ; a4 = address >> 16
0001029e  andi     a4,a4,0xff                     ; a4 = a[23:16]
        write_address_top
000102a2  addi     qspi_addr,a5,0x48              ; do{
000102a6  amoor.w  qspi_addr,a4,(qspi_addr)       ; qspi_txdata |= address[23:16]
000102aa  slli     a6,qspi_addr,0x20
000102ae  blt      a6,zero,write_address_top      ; } while(QSPI_TX_FIFO_FULL)
        addr_top_response
000102b2  c.lw     a4,0x4c(a5)                    ; do{
000102b4  blt      a4,zero,addr_top_response      ; } while(QSPI_RX_FIFO_EMPTY)
...
```

Дизассемблирование ZSBL. Анализ кода

Несоответствие программной и аппаратной части платы

```
...
        set_gpio
000106d8  lui       a5,0x10060
000106dc  addi      a3,a5,0xc
000106e0  c.lui     a4,0x8
000106e2  fence     0xf,0x5
000106e6  amoor.w.  zero,a4,(a3)  ; GPIO[15] = 1
000106ea  addi      a3,a5,0x8
000106ee  fence     0xf,0x5
000106f2  amoor.w.  zero,a4,(a3)  ; GPIO[15] as output
000106f6  addi      a5,a5,0x40
000106fa  fence     0xf,0x5
000106fe  amoor.w.  zero,a4,(a5)  ; invert GPIO[15]
...
```





Дизассемблирование ZSBL

Отображение результата выполнения инструкций в R2

Заключение

Извлечение данных

Анализируемые файлы:

- libr/arch/p/riscv/plugin.c — декомпиляция RISC-V инструкций
- libr/core/disasm.c — общая декомпиляция и вывод комментариев
- libr/include/r_anal.h — описание структуры RAnalOp, в которую сохраняется результат декомпиляции каждой инструкции
- libr/include/r_arch.h — описание структуры RArchValue, в которой хранятся операнды

```
RAnalOp *op = &ds->analop;
RArchValue *src = ((RArchValue *)(op->srcs.a));
```

Определение результата инструкции lui

```
lui    rd, imm[31:12]          ; Load upper immediate
```

Реализация в коде (файл disasm.c):

```
static void ds_print_determined_res(RDisasmState *ds) {
        ...

        if (r_str_startswith (op->mnemonic, "lui")) {
                imm = (src->imm << 12);
        }
        ...
}
```

Определение результата инструкции auipc

```
auipc       rd, imm[31:12]    ; Add upper immediate to pc
```

Реализация в коде (файл disasm.c):

```
static void ds_print_determined_res(RDisasmState *ds) {
...
RAnalFunction *f = fcnIn (ds, ds->analop.jump,
R_ANAL_FCN_TYPE_NULL);
      ...
      if (...) {
      ...
      } else if (r_str_startswith (op->mnemonic, "auipc")
                          && (!fcn || !fcn->name)) {
            imm = (src->imm << 12) + (op->addr);
      }
      ...
}
```

Результат модификации

```
0x000100e4      auipc    sp,     0x81d0
0x000100e8      addi     sp,     sp,  -228
0x000100ec      sub      sp,     sp,  t0
0x000100f0      jal      ra,     fcn.00010176
0x000100f4      lui      t0,     0x8000
0x000100f8      csrr     a0,     mhartid
0x000100fc      auipc    a1,     0x1
```

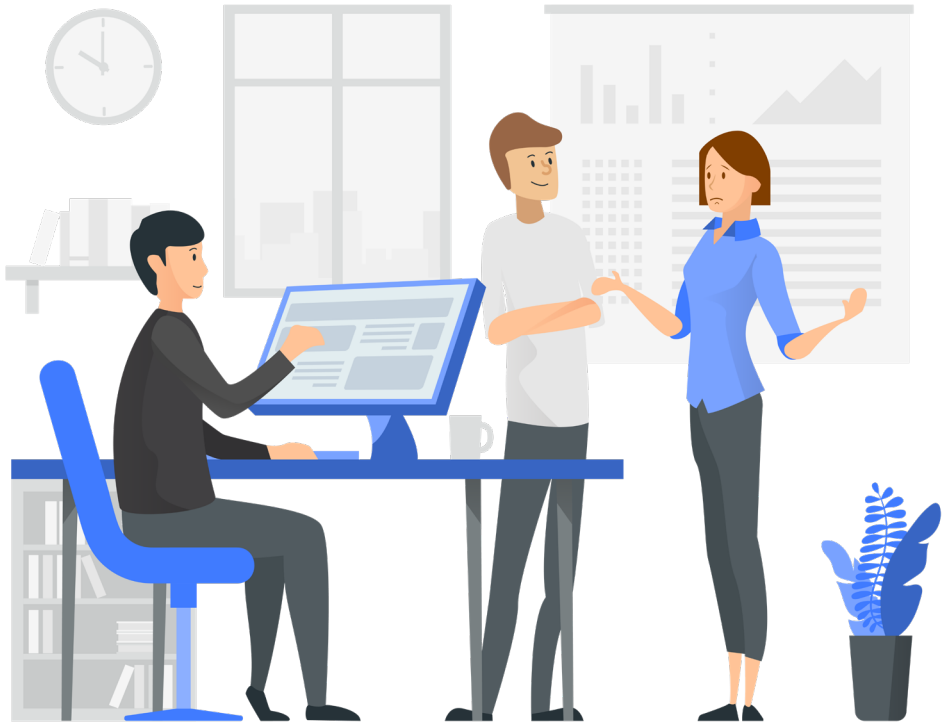
```
0x000100e4      auipc    sp,     0x81d0          ; 0x81e00e4
0x000100e8      addi     sp,     sp,  -228
0x000100ec      sub      sp,     sp,  t0
0x000100f0      jal      ra,     fcn.00010176
0x000100f4      lui      t0,     0x8000          ; 0x8000000
0x000100f8      csrr     a0,     mhartid
0x000100fc      auipc    a1,     0x1             ; 0x110fc
```

Дизассемблирование ZSBL

Отображение результата выполнения инструкций в R2

Заключение

Заключение



01 Настроена среда для совместной работы (сервер Ghidra);

02 Успешно выполнено дизассемблирование стартового кода с применением Ghidra;

03 Проведён анализ стартового кода;

04 Реализовано отображение результатов исполнения инструкций в Radare2;


БУДУЩЕЕ
В НАШИХ
РУКАХ