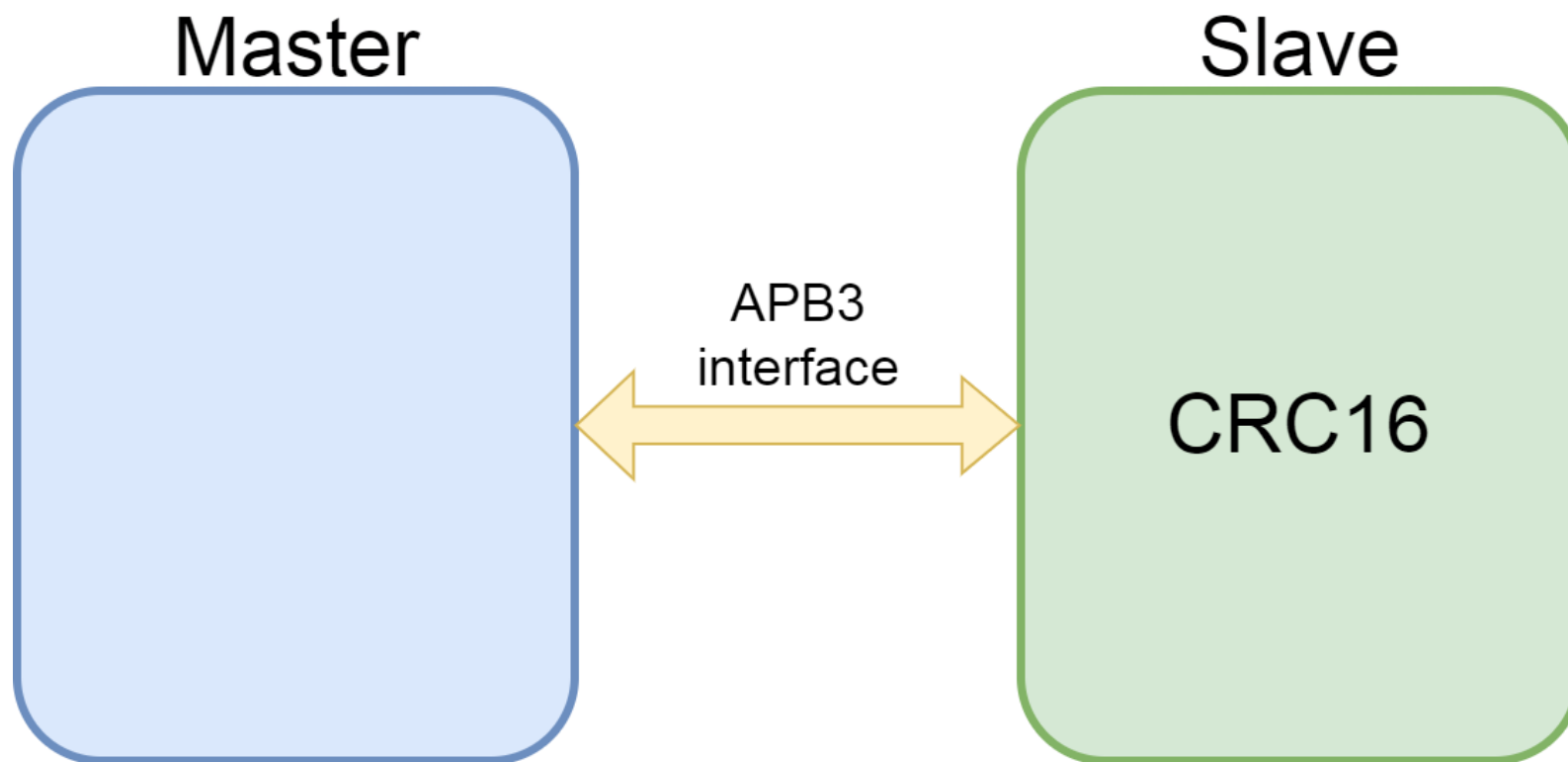


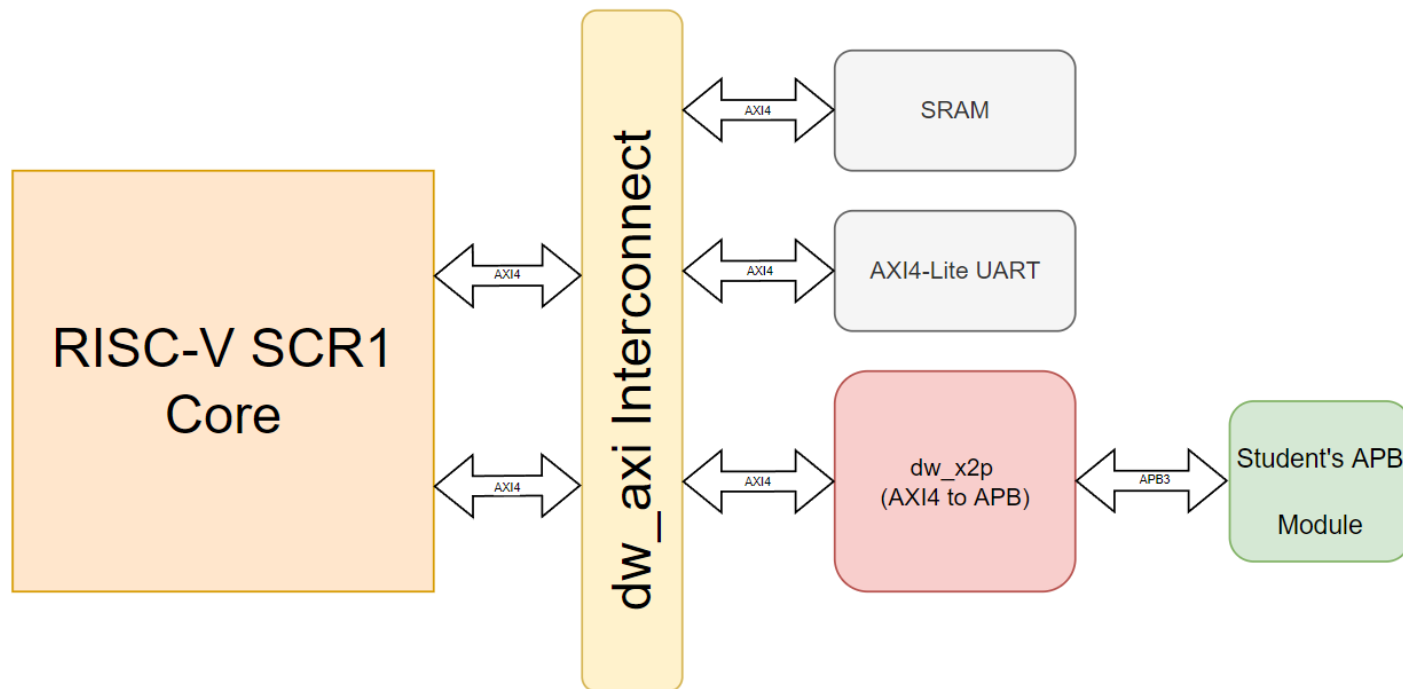
СИСТЕМЫ НА КРИСТАЛЛЕ (СНК)

Лабораторная работа №3

Блок вычисления CRC



Структура учебного проекта СнК



Указатели в С

Указатель – переменная, значением которой является адрес ячейки памяти. То есть указатель ссылается на блок данных из области памяти, причём на самое его начало. Указатель может ссылаться на переменную или функцию. Для этого нужно знать адрес переменной или функции.

Объявление указателей такой же, как и принцип объявления переменных. Отличие заключается только в том, что перед именем ставится символ звёздочки *. Чтобы получить значение, записанное в некоторой области, на которое ссылается указатель нужно воспользоваться операцией разыменования указателя *.

```
1  [type] * pointer_name = & ( variable );|
```

Синтаксис конструкции

```
1  int variable = 1;  
2  int* pointer = &variable;  
3  *pointer = 5;
```

Пример использования

Ссылки в С

Ссылки – особый тип данных, являющийся скрытой формой указателя, который при использовании автоматически разыменовывается. Ссылка может быть объявлена как другим именем, так и как псевдоним переменной, на которую ссылается.

При объявлении ссылки перед её именем ставится символ амперсанда &, сама же ссылка должна быть проинициализирована именем переменной, на которую она ссылается. Тип данных, на который указывает ссылка, может быть любым, но должен совпадать с объектом, на который ссылается, то есть с типом данных ссылочной переменной.

```
1  [type] & pointer_name = variable;
```

Синтаксис конструкции

```
1  int variable = 1;  
2  int& ref = variable;  
3  ref = 5;
```

Пример использования

Преобразование типов

Приведение типов в стиле языка C может привести выражение любого типа к любому другому типу данных (исключение это приведение пользовательских типов по значению, если не определены правила их приведения, а также приведение вещественного типа к указателю или наоборот). К примеру, `unsigned int` может быть преобразован к указателю на `double`.

```
1  [new type] variable;
```

Синтаксис конструкции

```
1  int variable_1 = 1;  
2  float variable_2 = ( float ) variable_1;
```

Пример использования

Struct

Структуры — это совокупность переменных, объединенных одним именем, предоставляющая общепринятый способ совместного хранения информации. Объявление структуры приводит к образованию шаблона, используемого для создания объектов структуры. Переменные, образующие структуру, называются членами структуры. Члены структуры также часто называются элементами или полями.

```
1  #include <stdint.h>
2
3  #define APB3_DEVICE_BASE_ADDRESS    0x400
4
5  typedef struct {
6      uint32_t    input_data;
7      uint32_t    result;
8      uint32_t    flags;
9      uint32_t    status;
10 } my_device_t;
11
12
13 void main() {
14
15     volatile my_device_t* my_device = (my_device_t*) APB3_DEVICE_BASE_ADDRESS;
16
17 }
```

Библиотека переменных `stdint.h`

Заголовочный файл ***stdint.h*** описывает целочисленные типы данных с установленными диапазонами представления чисел. Вместе с типами данных, в этом файле определены макросы с указанием верхних и нижних границ представляемых значений и макро-функции для формирования диапазонов представляемых значений для каждого типа данных.

Знаковый	Без знаковый	Описание
<code>int_8t</code>	<code>uint_8t</code>	Целочисленные типы данных с шириной диапазона представления чисел 8, 16, 32 и 64 бита, соответственно.
<code>int_16t</code>	<code>uint_16t</code>	
<code>int_32t</code>	<code>uint_32t</code>	
<code>int_64t</code>	<code>uint_64t</code>	

Ключевое слово *volatile*

Volatile — ключевое слово языков C/C++, которое информирует компилятор о том, что значение переменной может меняться из вне и что компилятор не будет оптимизировать эту переменную.

Объекты, объявленные как *volatile*, не используются в определенных оптимизациях, так как их значения могут изменяться в любое время.

При запросе объекта с ключевым словом *volatile* система всегда считывает его текущее значение, даже если оно запрашивалось в предшествовавшей инструкции. Кроме того, значение объекта записывается непосредственно при присваивании.

```
1 volatile declarator ;
```

```
1 volatile uint32 * statusPtr = 0xF1230000;
```

Синтаксис конструкции

Пример использования

Здесь **statusPtr** указывает на участок памяти, который в любой момент может быть перезаписан. Наша программа, в которой объявлен и проинициализирован этот указатель, не знает, когда это может произойти. От нее тут ничего не зависит. Но благодаря ключевому слову *volatile* можно надеяться, что при каждом обращении по этому адресу мы будем получать актуальное изменяемое значение.

Цели лабораторной работы

- 1) Подключить блок расчета CRC из лабораторной работы №2 к СнК по APB3 интерфейсу;
- 2) Написать программу на С, в которой происходит подключение к блоку CRC и считывание флага завершения расчета.

Выполнение Л/Р

1. Подключить компилятор. С помощью команды `nedit ~/.bashrc` добавить в файл

```
export PATH=/local/pkims06/SOC_PROGRAMMING_ARCH/riscv-gcc-10.2.0/bin:$PATH
```

2. Скопировать проект СнК в свою локальную директорию.

Проект находится по следующему пути:

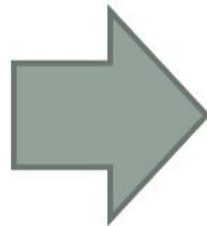
```
/local/pkims06/SOC_PROGRAMMING_ARCH/soc_programming
```

3. Скопировать файлы из Л/Р2 в директорию

```
<your_path>/soc_programming/src/rtl
```

4. Подключение блока к топ-уровню СнК

```
137
138 //|-----
139 //| Student's module place
140 //|-----
141
142
143 //| !!!!!!!!!!!!!!!!!!!!!
144 //| !!!!!!!!!!!!!!!!!!!!!
145 //| !!!!!!!!!!!!!!!!!!!!!
146 //| !!!!!!!!!!!!!!!!!!!!!
147 //| PUT YOUR MODULE HERE
148 //| !!!!!!!!!!!!!!!!!!!!!
149 //| !!!!!!!!!!!!!!!!!!!!!
150 //| !!!!!!!!!!!!!!!!!!!!!
151 //| !!!!!!!!!!!!!!!!!!!!!
152 //| !!!!!!!!!!!!!!!!!!!!!
153
```



```
143 APB #(
144     .ADDR_WIDTH      ( APB3_ADDR_WIDTH  ),
145     .DATA_WIDTH      ( APB3_DATA_WIDTH  )
146 ) APB3_sec (
147     .PCLK             ( i_clk           ),
148     .PRESETn          ( i_rst_n        ));
149
150 APB_slave#(
151     .ADDR_WIDTH      ( APB3_ADDR_WIDTH  ),
152     .DATA_WIDTH      ( APB3_DATA_WIDTH  )
153 ) crc_apb_slave (
154     .APB_if          ( APB3_sec.Slave  ));
155
156 always_comb APB3_sec.PADDR      = APB3.PADDR;
157 always_comb APB3_sec.PSEL       = APB3.PSEL;
158 always_comb APB3_sec.PENABLE   = APB3.PENABLE;
159 always_comb APB3_sec.PWRITE    = APB3.PWRITE;
160 always_comb APB3_sec.PWDATA    = APB3.PWDATA;
161
162 always_comb APB3.PREADY        = APB3_sec.PREADY;
163 always_comb APB3.PSLVERR      = APB3_sec.PSLVERR;
164 always_comb APB3.PRDATA       = APB3_sec.PRDATA;
```

5. Написание структуры доступа к CRC

<your_path>/soc_programming/src/compile/user_programm/main.c

```
1  #include <stdint.h>
2
3  #define APB3_DEVICE_BASE_ADDRESS      0x400
4
5  typedef struct {
6      uint32_t    input_data;
7      uint32_t    result;
8      uint32_t    flags;
9      uint32_t    status;
10 } my_device_t;
11
12
13 void main() {
14
15     volatile my_device_t* my_device = (my_device_t*) APB3_DEVICE_BASE_ADDRESS;
16
17 }
```

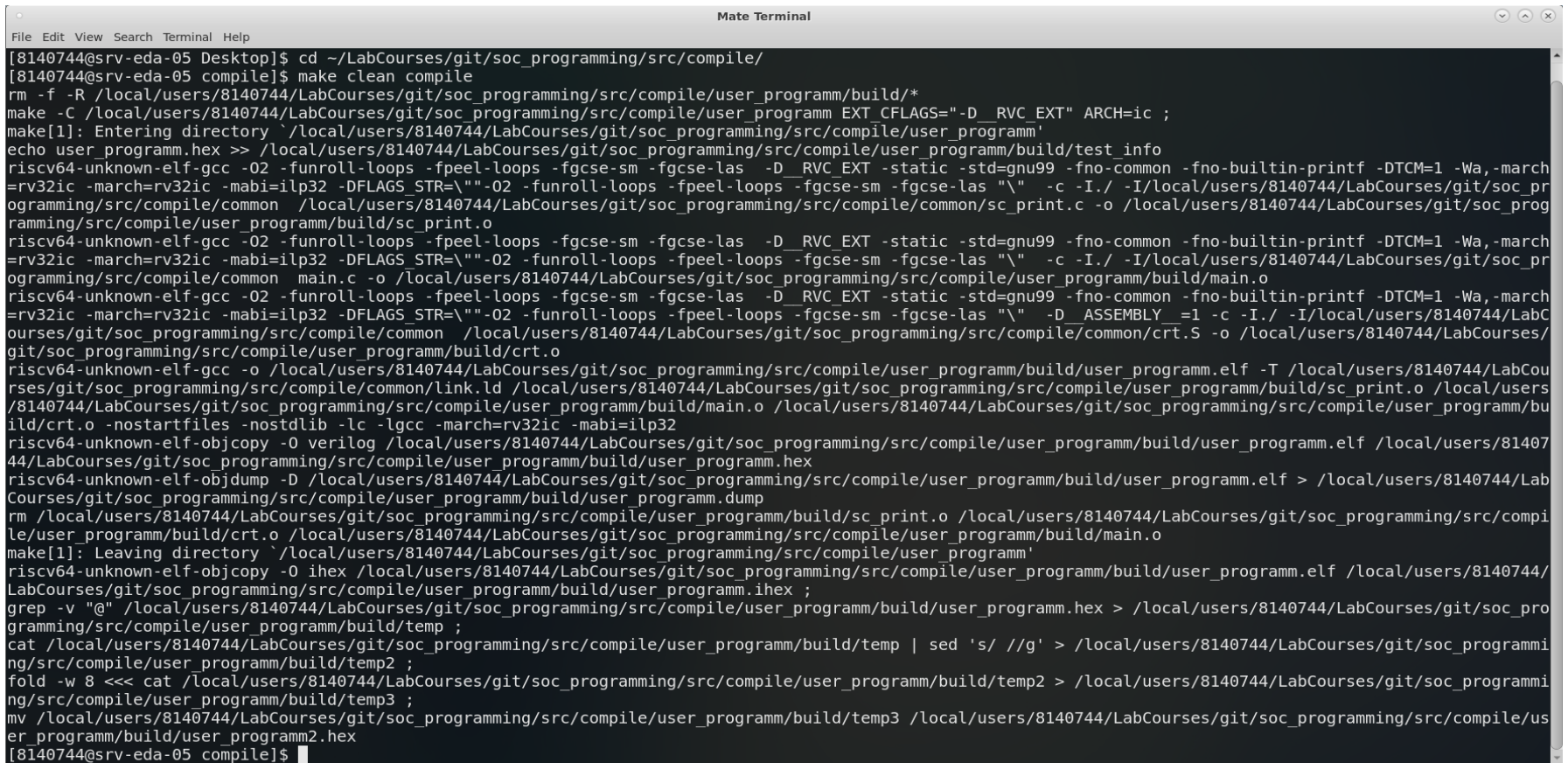
6. Написание кода на C

<your_path>/soc_programming/src/compile/user_programm/main.c

```
26 void main() {
27
28     volatile my_device_t* my_device = (my_device_t*) APB3_DEVICE_BASE_ADDRESS;
29
30     uint32_t    temp_var;
31
32     my_device->input_data    = 0xbaadc0de;
33     my_device->flags         = 0x1;
34
35
36     //Логика считывания флага завершения расчета CRC
37     //Логика записи в статусный регистр 0xFFFF при правильном расчете и 0xBEDA в ином
38
39 }
```

7. Компиляция кода

```
cd <your_path>/soc_programming/src/compile/  
make clean compile
```



```
Mate Terminal  
[8140744@srv-eda-05 Desktop]$ cd ~/LabCourses/git/soc_programming/src/compile/  
[8140744@srv-eda-05 compile]$ make clean compile  
rm -f -R /local/users/8140744/LabCourses/git/soc_programming/src/compile/user_programm/build/*  
make -C /local/users/8140744/LabCourses/git/soc_programming/src/compile/user_programm EXT_CFLAGS="-D_RVC_EXT" ARCH=ic ;  
make[1]: Entering directory `/local/users/8140744/LabCourses/git/soc_programming/src/compile/user_programm'  
echo user_programm.hex >> /local/users/8140744/LabCourses/git/soc_programming/src/compile/user_programm/build/test_info  
riscv64-unknown-elf-gcc -O2 -funroll-loops -fpeel-loops -fgcse-sm -fgcse-las -D_RVC_EXT -static -std=gnu99 -fno-common -fno-builtin-printf -DTCM=1 -Wa,-march=rv32ic -march=rv32ic -mabi=ilp32 -DFLAGS_STR="" -O2 -funroll-loops -fpeel-loops -fgcse-sm -fgcse-las "" -c -I./ -I/local/users/8140744/LabCourses/git/soc_programming/src/compile/common /local/users/8140744/LabCourses/git/soc_programming/src/compile/common/sc_print.c -o /local/users/8140744/LabCourses/git/soc_programming/src/compile/user_programm/build/sc_print.o  
riscv64-unknown-elf-gcc -O2 -funroll-loops -fpeel-loops -fgcse-sm -fgcse-las -D_RVC_EXT -static -std=gnu99 -fno-common -fno-builtin-printf -DTCM=1 -Wa,-march=rv32ic -march=rv32ic -mabi=ilp32 -DFLAGS_STR="" -O2 -funroll-loops -fpeel-loops -fgcse-sm -fgcse-las "" -c -I./ -I/local/users/8140744/LabCourses/git/soc_programming/src/compile/common main.c -o /local/users/8140744/LabCourses/git/soc_programming/src/compile/user_programm/build/main.o  
riscv64-unknown-elf-gcc -O2 -funroll-loops -fpeel-loops -fgcse-sm -fgcse-las -D_RVC_EXT -static -std=gnu99 -fno-common -fno-builtin-printf -DTCM=1 -Wa,-march=rv32ic -march=rv32ic -mabi=ilp32 -DFLAGS_STR="" -O2 -funroll-loops -fpeel-loops -fgcse-sm -fgcse-las "" -D_ASSEMBLY=1 -c -I./ -I/local/users/8140744/LabCourses/git/soc_programming/src/compile/common /local/users/8140744/LabCourses/git/soc_programming/src/compile/common/crt.S -o /local/users/8140744/LabCourses/git/soc_programming/src/compile/user_programm/build/crt.o  
riscv64-unknown-elf-gcc -o /local/users/8140744/LabCourses/git/soc_programming/src/compile/user_programm/build/user_programm.elf -T /local/users/8140744/LabCourses/git/soc_programming/src/compile/common/link.ld /local/users/8140744/LabCourses/git/soc_programming/src/compile/user_programm/build/sc_print.o /local/users/8140744/LabCourses/git/soc_programming/src/compile/user_programm/build/main.o /local/users/8140744/LabCourses/git/soc_programming/src/compile/user_programm/build/crt.o -nostartfiles -nostdlib -lc -lgcc -march=rv32ic -mabi=ilp32  
riscv64-unknown-elf-objcopy -O verilog /local/users/8140744/LabCourses/git/soc_programming/src/compile/user_programm/build/user_programm.elf /local/users/8140744/LabCourses/git/soc_programming/src/compile/user_programm/build/user_programm.hex  
riscv64-unknown-elf-objdump -D /local/users/8140744/LabCourses/git/soc_programming/src/compile/user_programm/build/user_programm.elf > /local/users/8140744/LabCourses/git/soc_programming/src/compile/user_programm/build/user_programm.dump  
rm /local/users/8140744/LabCourses/git/soc_programming/src/compile/user_programm/build/sc_print.o /local/users/8140744/LabCourses/git/soc_programming/src/compile/user_programm/build/crt.o /local/users/8140744/LabCourses/git/soc_programming/src/compile/user_programm/build/main.o  
make[1]: Leaving directory `/local/users/8140744/LabCourses/git/soc_programming/src/compile/user_programm'  
riscv64-unknown-elf-objcopy -O ihex /local/users/8140744/LabCourses/git/soc_programming/src/compile/user_programm/build/user_programm.elf /local/users/8140744/LabCourses/git/soc_programming/src/compile/user_programm/build/user_programm.ihex ;  
grep -v "e" /local/users/8140744/LabCourses/git/soc_programming/src/compile/user_programm/build/user_programm.hex > /local/users/8140744/LabCourses/git/soc_programming/src/compile/user_programm/build/temp ;  
cat /local/users/8140744/LabCourses/git/soc_programming/src/compile/user_programm/build/temp | sed 's/ //g' > /local/users/8140744/LabCourses/git/soc_programming/src/compile/user_programm/build/temp2 ;  
fold -w 8 <<< cat /local/users/8140744/LabCourses/git/soc_programming/src/compile/user_programm/build/temp2 > /local/users/8140744/LabCourses/git/soc_programming/src/compile/user_programm/build/temp3 ;  
mv /local/users/8140744/LabCourses/git/soc_programming/src/compile/user_programm/build/temp3 /local/users/8140744/LabCourses/git/soc_programming/src/compile/user_programm/build/user_programm2.hex  
[8140744@srv-eda-05 compile]$
```

8. Запуск симуляции

1. `cd <your_path>/soc_programming/src/compile/`

2. `module load synopsys/VCS/R-2020.12`

3. `make clean sim`