Лабораторная работа №5.

Прерывания.

Цель работы. Исследование процесса аппаратного прерывания. Изучение системы настроек контроллера прерываний GIC.

Задание на работу в лаборатории.

Изучить структуру предложенного комплекса программ. Набрать коды программ в текстовом редакторе, загрузить в макет DE1-SoC, откомпилировать и запустить. Нажимая поочередно кнопки макета, пронаблюдать результаты работы программы.

Программа 1.

```
.text

.global _start

B _start // reset vector
B SERVICE_UND // undefined instruction vector
B SERVICE_SVC // software interrrupt vector
B SERVICE_ABT_INST // aborted prefetch vector
B SERVICE_ABT_DATA // aborted data vector
.word 0 // unused vector
B SERVICE_IRQ // IRQ interrupt vector
B SERVICE_FIQ // FIQ interrupt vector
_start:
MOV R1, #0b11010010   // interrupts masked, MODE = IRQ
MSR CPSR_c, R1   // change to IRQ mode
LDR SP, =0xFFFFFFFF – 3   // set IRQ stack to A9 onchip memory
MOV R1, #0b11010011   // interrupts masked, MODE = SVC
MSR CPSR, R1   // change to supervisor mode
LDR SP, =0x3FFFFFFF – 3   // set SVC stack to top of DDR3 memory
BL CONFIG_GIC   // configure the ARM GIC
LDR R0, =0xFF200050   // pushbutton KEY base address
MOV R1, #0xF   // set interrupt mask bits
STR R1, [R0, #0x8]   // interrupt mask register (base + 8)
MOV R0, #0b01010011   // IRQ unmasked, MODE = SVC
MSR CPSR_c, R0
IDLE:
B IDLE // main program simply idles
```

```
SERVICE_UND:
B SERVICE_UND

SERVICE_SVC:
B SERVICE_SVC

SERVICE_ABT_DATA:
B SERVICE_ABT_DATA

SERVICE_ABT_INST:
B SERVICE_ABT_INST

SERVICE_IRQ:
PUSH {R0-R7, LR}
LDR R4, =0xFFFEC100
LDR R5, [R4, #0x0C] // read from ICCIAR
FPGA_IRQ1_HANDLER:
CMP R5, #73
UNEXPECTED:
BNE UNEXPECTED // if not recognized, stop here
BL KEY_ISR     //
EXIT_IRQ:  //
STR R5, [R4, #0x10] // write to ICCEOIR
POP {R0-R7, LR}
SUBS PC, LR, #4

SERVICE_FIQ:
B SERVICE_FIQ
.end // !!!  (3)
```

Программа 2.

```
.text

.global CONFIG_GIC

CONFIG_GIC:     //

PUSH {LR}
MOV R0, #73 // KEY port (Interrupt ID = 73)
MOV R1, #1 // this field is a bit-mask; bit 0 targets cpu0
BL CONFIG_INTERRUPT     //
LDR R0, =0xFFFEC100 // base address of CPU Interface
LDR R1, =0xFFFF // enable interrupts of all priorities levels
STR R1, [R0, #0x04]
MOV R1, #1
```

```
STR R1, [R0]
LDR R0, =0xFFFED000
STR R1, [R0]
POP {PC}   //

CONFIG_INTERRUPT:    //
PUSH {R4-R5, LR}
LSR R4, R0, #3 // calculate reg_offset
BIC R4, R4, #3 // R4 = reg_offset
LDR R2, =0xFFFED100
ADD R4, R2, R4 // R4 = address of ICDISER
AND R2, R0, #0x1F // N mod 32
MOV R5, #1 // enable
LSL R2, R5, R2 // R2 = value
LDR R3, [R4] // read current register value
ORR R3, R3, R2 // set the enable bit
STR R3, [R4] // store the new register value
BIC R4, R0, #3 // R4 = reg_offset
LDR R2, =0xFFFED800
ADD R4, R2, R4 // R4 = word address of ICDIPTR
AND R2, R0, #0x3 // N mod 4
ADD R4, R2, R4 // R4 = byte address in ICDIPTR
STRB R1, [R4]
POP {R4-R5, PC}   //

.end
```

Программа 3.

```
.text

.global KEY_ISR
KEY_ISR:      //
LDR R0, =0xFF200050 // base address of pushbutton KEY port
LDR R1, [R0, #0xC] // read edge capture register
MOV R2, #0xF
STR R2, [R0, #0xC] // clear the interrupt
LDR R0, =0xFF200020 // based address of HEX display
CHECK_KEY0:
MOV R3, #0x1
ANDS R3, R3, R1 // check for KEY0
BEQ CHECK_KEY1
MOV R2, #0b00111111
STR R2, [R0] // display "0"
```

```
B END_KEY_ISR
CHECK_KEY1:
MOV R3, #0x2
ANDS R3, R3, R1 // check for KEY1
BEQ CHECK_KEY2
MOV R2, #0b00000110
STR R2, [R0] // display "1"
B END_KEY_ISR
CHECK_KEY2:
MOV R3, #0x4
ANDS R3, R3, R1 // check for KEY2
BEQ IS_KEY3
MOV R2, #0b01011011
STR R2, [R0] // display "2"
B END_KEY_ISR
IS_KEY3:
MOV R2, #0b01001111
STR R2, [R0] // display "3"
END_KEY_ISR:
BX LR //
.end
```

Отчет должен содержать коды используемых программ с пояснением хода выполнения.