Міністерство освіти та науки України
Національний технічний університет України
«Київський політехнічний інститут ім. Ігоря Сікорського»
Факультет інформатики і обчислювальної техніки
Кафедра обчислювальної техніки

Лабораторна робота №4

з курсу «Архітектура комп’ютерів – 3»

Виконав:
студент 3 курсу ФІОТ
групи ІВ-81 :
Бухтій О. В. , 8106

Київ-2020

Виконання роботи:

$8106_{10}=1111110101010_2$

(0010) F = 2(X1 + X2 + X3) - (X4 + 1)/4

(10) X1 = 18, X2 = -9, X3 = 23, X4 = 11

Крок	Перевірити умови
1	*Перевірити ознаку результату (F) переповнення за межі розрядної сітки, якщо було переповнення виконати корекцію результату шляхом зсуву результату праворуч і запису значення переповнення в старший розряд, якщо ні, перейти на крок 2*
2	*Перевірити на рівність нулю молодшої частини результату [Ry] = 0 (якщо так перейти на крок 3, інакше виконати крок 4*
3	*Записати номер залікової книжки в старші біти молодшої частини числа (наприклад значення 9301 перевести в двійкову систему числення) [Ry]:=№ЗК, перейти на крок 4*
4	*Додати до старшої частини результату номер залікової книжки (наприклад значення 9301 подане в двійковій системі числення, [Rx];=[Rx]+№ЗК) перейти на крок 5*
5	*Визначити, чи є порушення нормалізації ліворуч (знакові розряди не збігаються), вважаємо, що результат поданий в модифікованому коді – два старші розряди старшої частини результату [Rx]- знакові*
6	*Виконати корекцію результату шляхом зсуву праворуч на один розряд*
7	*Перевірити ознаку переповнення в знакові розряди, якщо немає записати результат в пам’ять.*

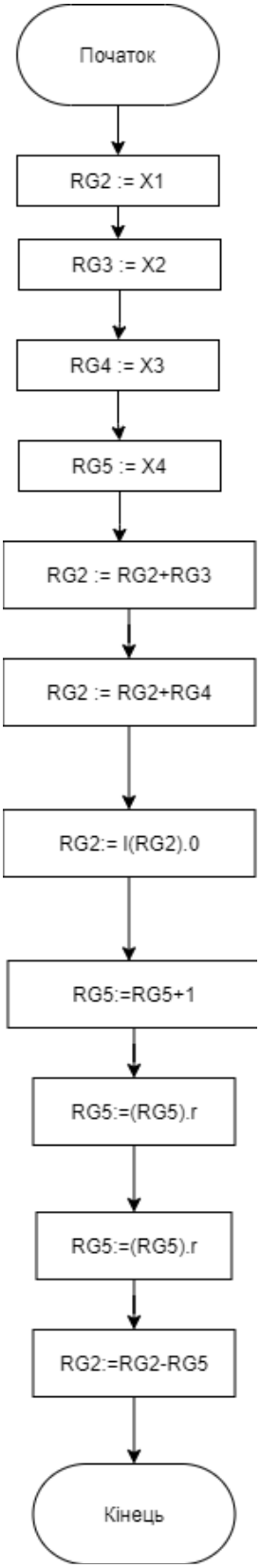
Алгоритм визначення коректності результату

(100) Номер точки переходу на підпрограму: V

(10) Номер точки переходу на підпрограму: XI

(00) Ознаки результату/умови переходу: Z=1, N=0, C=1

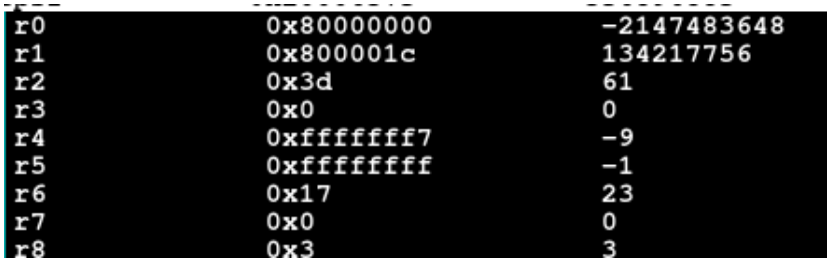
Алгоритм 1 завдання :



Висновок :

В цілому лабораторна робота ускладнена тільки тим що ми працюємо з подвійним словом і потрібно виконувати операції над двома регістрами як над одним тож код виглядає не зовсім чудово , але логіка зберігається тож і результат виходить очікуваним.

Скріни :



Результат виконання при використанні значень з варіанту. (в r2, r3 зберігався перше введене число, туди ж і збережено результат обчислень)

Лістинг :

---lscript.ld---

```
MEMORY
{
     FLASH ( rx )    : ORIGIN = 0x08000000, LENGTH = 1M
     RAM ( rxw )     : ORIGIN = 0x20000000, LENGTH = 128K
}
__stack_start = ORIGIN(RAM) + LENGTH(RAM);

/*ENTRY(__hard_reset__);*/

SECTIONS
{
    .text :
    {
        . = ALIGN(4);
        KEEP(*(.interrupt_vector))
        *(.text)
        *(.text*)
        *(.rodata)
        *(.rodata*)
        . = ALIGN(4);
    } > FLASH
}
```

---Makefile---

```
PATH := ${PATH}:/opt/xPacks/qemu-arm/2.8.0-12/bin:\
/opt/gcc-arm-9.2-2019.12-x86_64-arm-none-eabi/bin

SDK_PREFIX?=arm-none-eabi-
CC = $(SDK_PREFIX)gcc
LD = $(SDK_PREFIX)ld
SIZE = $(SDK_PREFIX)size
OBJCOPY = $(SDK_PREFIX)objcopy
QEMU = qemu-system-gnuarmeclipse
BOARD ?= STM32F4-Discovery
MCU=STM32F407VG
TARGET=firmware
CPU_CC=cortex-m4
TCP_ADDR=1234
#################
# CFLAGS
CFLAGS = -O0 -g3 -Wall
###################
# LDFLAGS
LDFLAGS = -Wall --specs=nosys.specs -nostdlib -lgcc
###################
# PATH
APP_PATH=$(abspath ./)
###################
# add here GNU ASSEMBLY SOURCES .S
GASSRC += print.S
GASSRC += start.S
```

```
    28     ##################
    29     SOBJS = $(GASSRC:.S=.o)
    30     COBJS = $(patsubst .c,%.o,$(APP_SRC))
    31     .PHONY: all clean
    32     # Path to directories containing application source
    33     vpath % $(APP_PATH)
    34
    35
    36     #-----------------------------------
Build-------------------------------
    37     all: $(SOBJS) $(TARGET).elf $(TARGET).bin
    38
    39     #object_files
    40     %.o: %.S
    41          $(CC) -g -x assembler-with-cpp  $(CFLAGS) -mcpu=$(CPU_CC) -
c -o $@ $^
    42
    43
    44     #executable_&_linkable_file
    45     $(TARGET).elf: $(SOBJS)
    46          $(CC) $^ -mcpu=$(CPU_CC) $(LDFLAGS) -T./lscript.ld -o $@
    47
    48     #binary_file
    49     $(TARGET).bin: $(TARGET).elf
    50          $(OBJCOPY) -O binary -F elf32-littlearm $(TARGET).elf $
(TARGET).bin
    51
    52     #-----------------------------------
QEMU--------------------------------
    53     qemu:
    54          $(QEMU)  --verbose --verbose --board $(BOARD) --mcu $(MCU)
-d unimp,guest_errors --image $(TARGET).elf --semihosting-config
enable=on,target=native -gdb tcp::$(TCP_ADDR) -S
    55     qemu_run:
    56          $(QEMU)  --verbose --verbose --board $(BOARD) --mcu $(MCU)
-d unimp,guest_errors --image $(TARGET).bin --semihosting-config
enable=on,target=native
    57
    58     clean:
    59          -rm *.o
    60          -rm *.elf
    61          -rm *.bin
```

---start.S---

```
     1     .syntax unified
     2     .cpu cortex-m4
     3     .thumb
     4
     5     .global vtable
     6     .global __hard_reset__
     7     .extern printf
     8
     9     .type vtable, %object
    10     .type __hard_reset__, %function
```

```
    11
    12      .section .interrupt_vector
    13      vtable:
    14           .word __stack_start
    15           .word __hard_reset__+1
    16           .size vtable, .-vtable
    17
    18      .section .rodata
    19           data: .asciz "lab4_P1 started!\n"
    20           endofprogram: .asciz "Done."
    21
    22           X1: .quad 0x0000000080000000
    23           X2: .quad 0x0000000080000000
    24           X3: .quad 0x0000000080000000 //для ДК змінить знаки, додать
1
    25           X4: .quad 0x0000000080000000 //<-- (-24) ДК (доповняльний
код)
    26           RS: .quad 0x0000000000000000
    27
    28      .section .text
    29
    30      __hard_reset__:
    31           bl part1
    32           ldr r0, =endofprogram
    33           bl dbgput_line
    34           end:
    35                b end
    36           ;.size __hard_reset__, .-__hard_reset__
    37
    38
    39      part1:
    40           push {lr}
    41           ldr r0, =data
    42           bl dbgput_line
    43
    44           ldr r0, =X1
    45           ldm r0, {r2, r3} //r2 - молодші, r3 - старші
    46
    47           ldr r0, =X2
    48           ldm r0, {r4, r5} //аналогічно
    49
    50           ldr r0, =X3
    51           ldm r0, {r6, r7}
    52
    53           ldr r0, =X4
    54           ldm r0, {r8, r9}
    55
    56
    57
    58           adds r2, r4 // X1+X2
    59           adc r3,    r5
    60
    61           adds r2, r6 //(X1+X2)+X3
    62           adc r3, r7
```

```
 63
 64          lsls r2, #1  // 2*(X1+X2+X3)
 65          lsl r3, #1
 66          IT CS
 67          addcs r3, #1 //if carry than add 1 to shifted reg
 68
 69
 70
 71          adds r8, #1 // X4+1
 72          adc r9,    #0
 73
 74          asrs r9, #1
 75          lsr r8, #1  // (X4+1)/2
 76          ldr r0, =0x80000000
 77          IT CS
 78          orrcs r8, r0 //if carry than shift in 1 from other side
 79
 80          asrs r9, #1
 81          lsr r8, #1  // (X4+1)/4
 82          ldr r0, =0x80000000
 83          IT CS
 84          orrcs r8, r0
 85
 86
 87
 88          subs r2, r8 // P1-P2 = 2*(X1+X2+X3) - (X4+1)/4
 89          sbc r3,    r9
 90
 91          ITT VS
 92          nopvs
 93          blvs check
 94
 95          ITT VC
 96          nopvc
 97          blvc branch
 98
 99          pop {pc}
100
101
102     check:
103          asrs r3, #1
104          lsr r2, #1
105          ldr r0, =0x80000000
106          IT CS
107          orrcs r2, r0
108
109          orr r3, r0
110
111     branch:
112          cmp r2, #0
113
114          ITT EQ
115          nopeq
116          beq sethigher
```

```
117
118          ITT NE
119          nopne
120          bne addtohigher
121
122     sethigher:
123          ldr r0, =0xFD500000 //8106 in hex shifted
124          mov r2, r0
125
126     addtohigher:
127          ldr r0, =8106
128          add r3, r0
129
130     part5:
131     part6:
132          asrs r3, #1
133          lsr r2, #1
134          ldr r0, =0x80000000
135          IT CS
136          orrcs r2, r0
137
138     save:
139          //ITT VS
140          //ldr r0, =RS
141          //stm r0, {r2, r3}
142          //str r2, [r0]
143
144
145          bx lr
146
147
148     //---------------Підпрограма 1------------------
149     part21:
150          push {lr}
151          ldr r0, =0x2A
152          lsl r0, #2
153
154     strange_return1:
155          add r0, #1
156
157          IT CS
158          bcs branch1
159
160          IT CC
161          bcc branch2
162
163     branch1:
164          add r0, #1
165          b block2
166
167
168     branch2:
169          add r0, #2
170          lsl r0, #1
```

```
   171          b block2
   172
   173     block2:
   174
   175          IT VS
   176          bvs strange_return1
   177
   178          add r0, #1
   179          pop {pc}
   180     //-------------------------------------------
   181
   182     //---------------Підпрограма 2-----------------
   183     part22:
   184          push {lr}
   185          add r0, #1
   186
   187          IT VC
   188          bvc branch12
   189
   190          IT VS
   191          bvs branch22
   192
   193     branch12:
   194          add r0, #1
   195          IT VC
   196          bvc part22
   197
   198     branch22:
   199          mov r0, r2
   200          add r0, #1
   201          pop {pc}
   202
   203     //-------------------------------------------
   204
   205     //---------------Підпрограма 3-----------------
   206     part23:
   207          push {lr}
   208          add r0, #1
   209          add r0, #1
   210          pop {pc}
   211
   212     //-------------------------------------------
   213
```

---print.S---

```
     1     .thumb
     2     .syntax unified
     3     .cpu cortex-m4
     4     #define SEMIHOSTING_SYS_WRITE0  #0x04      //Syscall
     5     #define SEMIHOSTING #0xAB                  //Supervisor call
number
     6
     7     .section .data
```

```
 8         str_hex: .asciz "0xXXXXXXXX\n"
 //??????????????????????
 9
10     .text
11     .global dbgput_line
12     .global dbgput
13     .global newline
14     .global dbgput_num
15
16     // param: @str
17     dbgput:
18         push {lr}
19         mov r1, r0                                  //Arg to syscall
20         mov r0, SEMIHOSTING_SYS_WRITE0    //Pass syscall
21         bkpt SEMIHOSTING                         //Supervisor call
22         pop {pc}
23
24     _newline_sym: .asciz "\n\r"                     //Newline
25     .align 4
26     dbgput_line:
27         push {lr}
28         mov r1, r0                                  //Arg to syscall
29         mov r0, SEMIHOSTING_SYS_WRITE0    //Pass syscall
30         bkpt SEMIHOSTING                         //Supervisor call
31
32         ldr r1,=_newline_sym                    //Arg to syscall
33         mov r0, SEMIHOSTING_SYS_WRITE0    //Pass syscall
34         bkpt SEMIHOSTING                         //Supervisor call
35         pop {pc}
36     newline:
37         push {lr}
38         ldr r1,=_newline_sym                    //Arg to syscall
39         mov r0, SEMIHOSTING_SYS_WRITE0    //Pass syscall
40         bkpt SEMIHOSTING                         //Supervisor call
41         pop {pc}
42     dbgput_num:
43         push {lr}
44     //---------------------------------
45         mov r2, #9                                  //Other
syscall ?
46         mov r3, #0x0000000F                         //Mask
47         ldr r1, =str_hex                         //Mask ?
48
49         next:
50             push {r0}
51             and r0, r3
52             add r0, #48
53             cmp r0, #58
54             blo store
55             add r0, #7
56
57         store:
58             strb r0, [r1, r2]
59             pop {r0}
```

```
60              lsr r0, r0, #4
61              sub r2, #1
62              cmp r2, #2
63              bge next
64
65      //---------------------------------
66              ldr r1, =str_hex                    //Arg to syscall
67              mov r0, SEMIHOSTING_SYS_WRITE0      //Pass syscasll
68              bkpt SEMIHOSTING                    //Supervisor call
69              pop {pc}
```