НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ

«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ»

ФАКУЛЬТЕТ ІНФОРМАТИКИ І ОБЧИСЛЮВАЛЬНОЇ ТЕХНІКИ

КАФЕДРА ОБЧИСЛЮВАЛЬНОЇ ТЕХНІКИ

**Лабораторна робота №4**

*з дисципліни* ***«****Архітектура комп’ютерів 3****»***

Виконав:

студент 3 курсу

групи ІВ-81

Бєлов М.С.

Перевірив:

Нікольсбкий С.С.

Київ 2021 р.

Варіант лаб 4 8102 = 0b1111110100110

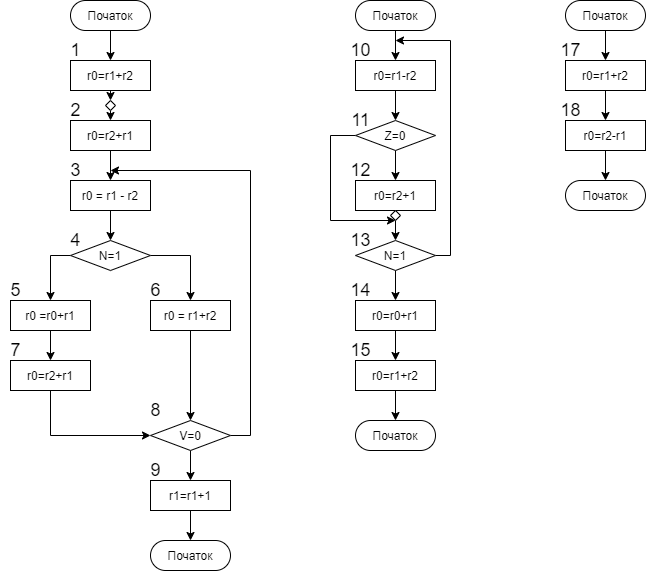
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 0 | 1 | 1 | 1 |  |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 1 | 0 | 18 | –9 | 23 | 11 |

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  | Номер точки переходу на підпрограму |
| 0 | 0 | 0 | І |

|  |  |  |
| --- | --- | --- |
| 1 | 0 | XI |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  | Умови переходу (ознаки результату виконання операції) | | |
|  |  | ***CD*1** | ***CD*2** | ***CD*3** |
| 0 | 1 | N=1 | Z=0 | V=1 |



**Лістинг програми:**

**lab4.S**

.global lab4

.syntax unified

.section .rodata

test\_first: .asciz "First under-program started."

test\_first\_2: .asciz "First under-program finished."

test\_sec: .asciz "Second under-program started."

test\_sec\_2: .asciz "Second under-program finished."

res\_f: .asciz "Result"

reg\_f: .asciz "First reg:\t"

reg\_s: .asciz "Second reg:\t"

.section .text

#define \_x1 #18

#define \_x2 #-9

#define \_x3 #23

#define \_x4 #11

// F = (X1 V X2 / 2) / 8 - 2 (not(X3) XOR 1)

lab4:

push {lr}

bl task1

bl task2

pop {pc}

task1:

push {lr}

// calculate

mov r0, \_x1

mov r1, #0

mov r2, \_x2

mov r3, #-1

mvn r4, \_x3

mov r5, #-1

mov r6, \_x4

mov r7, #0

bl calc

pop {pc}

calc:

push { lr }

// X2 / 2

asr r2, #1

asr r3, #1

// X1 v X2 / 2

orr r0, r2

orr r1, r3

// (X1 v X2 / 2) / 8

asr r0, #3

asr r1, #3

// not(X3) XOR 1

eor r4, #1

eor r5, #0

// (not(X3) XOR 1) \* 2

lsls r4, #1

lsl r5, #1

IT CS

orrCS r5, #1

// F

subs r0, r4

sbc r1, r5

mov r9, r0

mov r10, r1

ldr r0, = res\_f

bl dbgput\_line

ldr r0, = reg\_f

bl dbgput\_line

mov r0, r9

bl dbgput\_num

ldr r0, = reg\_s

bl dbgput\_line

mov r0, r1

mov r0, r10

bl dbgput\_num

mov r0, r9

mov r1, r10

bl start\_correction

pop { pc }

// Result correction

start\_correction:

push {lr}

mov r10, #8102

bvc step2

lsr r0, #1

lsr r1, #1

step2:

tst r0, #0

beq step4

//tst r1, #0

//be step4

mov r0, r10

step4:

add r1, r10

and r5, r1, #0xc0000000

beq step7

eor r5, #0xc0000000

beq step7

lsr r0, #1

lsr r1, #1

step7:

bvs end\_correction

push {r0}

push {r1}

pop {r1}

pop {r0}

end\_correction:

pop {pc}

task2:

push {lr}

mov r1, #25

mov r2, #56

program\_a:

// 1: r0 = r1 + r2

add r0, r1, r2

// jump to program b

bl program\_b

// 2: r0 = r2 + r1

add r0, r1, r2

t2\_loop1:

// 3: r0 = r1 - r2

subs r0, r1, r2

// 4: if N = 1

bmi t2\_loop1\_b1

t2\_loop1\_b0:

// 5: r0 = r0, r1

adds r0, r0, r1

// 7: r0 = r2 + r1

adds r0, r2, r1

b t2\_loop1\_check

t2\_loop1\_b1:

// 6: r0 = r1 + r2

adds r0, r1, r2

t2\_loop1\_check:

// 8: if V = 0

bvc t2\_loop1

// 9: r0 = r1 + 1

add r0, r1, #1

pop {pc}

program\_b:

push {lr}

push {r0, r1 }

ldr r0, =test\_first

bl dbgput\_line

pop {r0, r1 }

t2\_loop2:

// 10: r0 = r1 - r2

subs r0, r1, r2

// 11: if Z = 1 (inverted)

beq t2\_loop1\_b

// 12: r0 = r2 + 1

adds r0, r2, #1

// jump to program c

bl program\_c

t2\_loop1\_b:

bmi t2\_loop2

// 14: r0 = r0 + r1

adds r0, r0, r1

// 15: r0 = r1 + r2

add r0, r1, r2

push { r0, r1 }

ldr r0, =test\_first\_2

bl dbgput\_line

pop { r0, r1 }

pop {pc}

program\_c:

push {lr}

push {r0, r1 }

ldr r0, =test\_sec

bl dbgput\_line

pop {r0, r1 }

// 17: r0 = r1 + r2

add r0, r1, r2

// 18: r0 = r2 - r1

subs r0, r2, r1

push {r0, r1 }

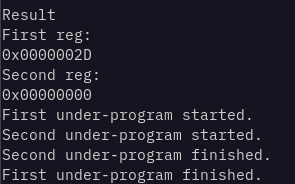
ldr r0, =test\_sec\_2

bl dbgput\_line

pop {r0, r1 }

pop {pc}

**Скріншоти**



**Висновок**

Під час виконання даної лабораторної роботи було створено, зібрано і протестовано на емуляторі проект з програмою на мові асемблера, яка обраховує результат заданої функції та використовує команди передавання управління.