/\*\*\*\*\*\*\*\*\*\*\*\*Assignment 3\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*C code\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

int32\_t lookup(float \*fArray, float f, uint32\_t size) //takes arguments fArray,f and size

{

int i;

for(i = 0; i < size; i++) //loop runs till the the size

{

if(fArray[i]==f) //compare each array element with f till it gets equal

return i; //if equal then return i

}

return -1; //return -1

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Assembly Code\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

.text

.global main

.type %function, main

main:

push {r4, lr} @store ip and lr to stack memory

ldr r1, =a @r1 <- &a / It initialize the array from its base address

B lookup @Branch to lookup / function call

lookup:

loop:

mov r2, #0 @r2 <- 0, i = 0;

cmp r2, #10 @Compare R2 with 10 / for loopsize = 10

bge done @Branch to done if r2 is greater or equal to 10 / for loop ends

mov r6,[r1,r2] @Move data from address of r1 and r2 to r6

cmp r6, r3 @farray(a[0]) == f(r3) / Compare array elements with r3

beq ret @branch to ret if equal

ret: mov r0, r2 @ return i

add r2, r2, #1 @r2 <- r2 + 1 / i++

b loop @Branch to loop / for loop

done:

mov r0, #-1 @return -1

pop {r4, pc} @store r4 and pc to stack memory

bx lr

.align 2

a: .size 40 @array of 10