**ARM ASSEMBLY COURSEWORK – 2**

CASSANDRA ANN FERNANDES

Bsc Computer Systems – Year 2

H00200702

This coursework was a challenging one to broaden our knowledge and application of arm assembly.

An initial array of 256 character spaces was allocated all set to 0 which stores the counter of each character that is present in the corresponding text file.

The index of the array acts as almost the ascii table that increments as long as there is a character in that position from the file.

The array is named text which gets the characters from the file stored in the variable =file. The \_test loop reads the file by taking the second argument from the executing in the terminal (./mycode mytext.txt) and reading it by one character each into the file variable.

It then checks if it is the end of file or if there are no more characters and branches to the closing of the file. If not empty then it starts to load the characters to a new pointer which acts as the ascii index of each character. And another register is pointed to the file contents which acts as a counter for the appropriate index of non zero elements.

The second branch is supposed to acts as a print function by comparing each index to see if it contains a counter and print it to the monitor.

References :

1. <https://vision.hw.ac.uk/bbcswebdav/pid-2004064-dt-content-rid-1686186_2/courses/F28HS_2015-2016/fshow.s>
2. <https://vision.hw.ac.uk/bbcswebdav/pid-2004064-dt-content-rid-1686196_2/courses/F28HS_2015-2016/seta.s>

@Cassandra Ann Fernandes

@BSC COMPUTER SYSTEMS

@YEAR 2 -- F28HS -- ARM ASSEMBLYCOURSEWORK --

.global \_start

\_start:

@ open input using argv[1]

LDR R0, [SP,#8] @argv[1]

MOV R1, #O\_RDONLY

BL open

LDR R1, =file @address of the file in R1

STR R0, [R1]

LDR R3, =text @text is array stored in R3

MOV R4, #4

\_test:

LDR R1, =file @loads address of file into r1

LDR R0,[R1] @pointer to contents of R1

LDR R1, =char @loaded address to R1; initialise R1 to 0

MOV R2, #1 @offset hhaa if 2

BL read

CMP R0, #0 @compares eof

BEQ endl

LDR R6, [R1] @loads the character to R6 which is 0

MUL R6, R4 @multip 4 to R6

ADD R3, R6 @ ascii

@LDR R4, [R3] @content of R3 IN R4

@ADD R3, #0x04

@ADD R4, #0x1

@STR R4, [R3]

LDR R5, [R3] @Loads contents of R3 to R5

ADD R5, #1 @increments index of ascii

STR R5, [R3] @store results of counter back to R3

SUB R3, R6 @goes back to original value of R3

B \_test

BL print

print:

CMP R3, #256

BLT endl

@MOV R0, #1

LDR R5, [R3]

STR R5, [R3]

ADD R5,#1

LDR R5, [R3]

ADD R5, #0

CMP R5, #0

BEQ print

ADD R3, #4

STR R5,[R3]

MOV R0,#1

LDR R1, =text

MOV R2, #1

BL write

@MOV R7, #4 @syscall for write

@SWI #0 @invoking syscall

endl:

@ close files

LDR R1, =file

LDR R0, [R1]

BL close

\_exit: MOV R7, #1

MOV R0, #0

SWI 0

read: MOV R7, #3

SWI 0

BX LR

write: MOV R7, #4

SWI 0

BX LR

open: MOV R7, #5

SWI 0

BX LR

close: MOV R7, #6

SWI 0

BX LR

.data

.equ O\_RDONLY, 0x0000

.equ O\_WRONLY, 0x0001

.equ O\_CREAT, 0x0200

.equ READP, 0x04

.equ WRITEP, 0x02

char: .word 0

file: .word 0

text: .rept 256

.word 0

.endr