Assembly Code

; \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

; File: COMPORGHW3.S

; Programmer: Manoj Vasa

; Class: CS 2400-002

; Description: Prints original string, analyzes each

; character for vowel, capitalizes lower case vowels, and

; prints altered string.

; Project: Vowel.arj.

; Date: Feb. 25, 2016

; \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

AREA Homework3, CODE

SWI\_WriteC EQU &0 ;output character in r0

SWI\_exit EQU &11 ;finish program

ENTRY

Main

ADR r1, String ;store string address in r1

ADR r2, String ;store string address in r2

MOV r3, #0 ;initialize counter

OriginalPrint LDRB r0, [r1], #1 ;load next byte

CMP r0, #0 ;check for null terminator of string

SWINE SWI\_WriteC ;if not equal, then print

BNE OriginalPrint ;loop back to OriginalPrint if not equal to 0

VowelCheck LDRB r0, [r2], #1 ;load next byte of string

CMP r0, #0 ;check for null terminator of string

BNE CheckSpace ;if not equal, go to CheckSpace

B CheckNum1

Exit SWI SWI\_exit

CheckSpace CMP r0, #32 ;check for space character in string

BNE CheckLower ;if not equal, check for lower case vowel

SWI SWI\_WriteC ;print space character if equal

B VowelCheck ;unconditinonal jump back to VowelCheck

CheckLower CMP r0, #96 ;compare byte to 96 to check for lower case vowel

BGE LowerVowela ;if greater than or equal to 96, go to LowerVowela

UpperVowelA CMP r0, #65 ;compare byte with decimal value of upper case A

BNE VowelE ;if not equal go to next vowel

ADDEQ r3, r3, #1 ;increment counter if byte is vowel

SWI SWI\_WriteC ;print character

B VowelCheck ;return to VowelCheck

VowelE CMP r0, #69 ;compare byte with decimal value of upper case E

BNE VowelI ;if not equal go to next vowel

ADDEQ r3, r3, #1 ;increment counter if byte is vowel

SWI SWI\_WriteC ;print character

B VowelCheck ;unconditinonal jump back to VowelCheck

VowelI CMP r0, #73 ;compare byte with decimal value of upper case I

BNE VowelO ;if not equal go to next vowel

ADDEQ r3, r3, #1 ;increment counter if byte is vowel

SWI SWI\_WriteC ;print character

B VowelCheck

VowelO CMP r0, #79 ;compare byte with decimal value of upper case O

BNE VowelU ;if not equal go to next vowel

ADDEQ r3, r3, #1 ;increment counter if byte is vowel

SWI SWI\_WriteC ;print character

B VowelCheck ;unconditinonal jump back to VowelCheck

VowelU CMP r0, #85 ;compare byte with decimal value of upper case U

BNE WriteCons ;if not equal go to WriteCons

ADDEQ r3, r3, #1 ;increment counter if byte is vowel

SWI SWI\_WriteC ;print character

B VowelCheck ;unconditinonal jump back to VowelCheck

LowerVowela CMP r0, #97 ;compare byte with decimal value of lower case a

BNE Vowele ;if not equal go to next vowel

ADDEQ r3, r3, #1 ;increment counter if byte is vowel

MOV r0, #65 ;replace a with A

SWI SWI\_WriteC ;print character

B VowelCheck ;unconditinonal jump back to VowelCheck

Vowele CMP r0, #101 ;compare byte with decimal value of lower case e

BNE Voweli ;if not equal go to next vowel

ADDEQ r3, r3, #1 ;increment counter if byte is vowel

MOV r0, #69 ;replace e with E

SWI SWI\_WriteC ;print character

B VowelCheck ;unconditinonal jump back to VowelCheck

Voweli CMP r0, #105 ;compare byte with decimal value of lower case i

BNE Vowelo ;if not equal go to next vowel

ADDEQ r3, r3, #1 ;increment counter if byte is vowel

MOV r0, #73 ;replace i with I

SWI SWI\_WriteC ;print character

B VowelCheck ;unconditinonal jump back to VowelCheck

Vowelo CMP r0, #111 ;compare byte with decimal value of lower case o

BNE Vowelu ;if not equal go to next vowel

ADDEQ r3, r3, #1 ;increment counter if byte is vowel

MOV r0, #79 ;replace o with O

SWI SWI\_WriteC ;print character

B VowelCheck ;unconditinonal jump back to VowelCheck

Vowelu CMP r0, #117 ;compare byte with decimal value of lower case u

BNE WriteCons ;if not equal go to WriteCons

ADDEQ r3, r3, #1 ;increment counter if byte is vowel

MOV r0, #85 ;replace u with U

SWI SWI\_WriteC ;print character

B VowelCheck ;unconditinonal jump back to VowelChecks

WriteCons SWI SWI\_WriteC ;print character

B VowelCheck ;unconditinonal jump back to VowelCheck

CheckNum1 CMP r3, #1

BNE CheckNum2

ADR r8, StringNum1 ;load string

printNum1 LDRB r0, [r8], #1 ;load counter into r0 if r0 equal to 0

CMP r0, #0 ;check null terminator

SWINE SWI\_WriteC ;print r0

BNE printNum10

B Exit

CheckNum2 CMP r3, #2

BNE CheckNum3

ADR r8, StringNum2 ;load string

printNum2 LDRB r0, [r8], #1 ;load counter into r0 if r0 equal to 0

CMP r0, #0 ;check null terminator

SWINE SWI\_WriteC ;print r0

BNE printNum2

B Exit

CheckNum3 CMP r3, #3

BNE CheckNum4

ADR r8, StringNum3 ;load string

printNum3 LDRB r0, [r8], #1 ;load counter into r0 if r0 equal to 0

CMP r0, #0 ;check null terminator

SWINE SWI\_WriteC ;print r0

BNE printNum3

B Exit

CheckNum4 CMP r3, #4

BNE CheckNum5

ADR r8, StringNum4 ;load string

printNum4 LDRB r0, [r8], #1 ;load counter into r0 if r0 equal to 0

CMP r0, #0 ;check null terminator

SWINE SWI\_WriteC ;print r0

BNE printNum4

B Exit

CheckNum5 CMP r3, #5

BNE CheckNum6

ADR r8, StringNum5 ;load string

printNum5 LDRB r0, [r8], #1 ;load counter into r0 if r0 equal to 0

CMP r0, #0 ;check null terminator

SWINE SWI\_WriteC ;print r0

BNE printNum5

B Exit

CheckNum6 CMP r3, #6

BNE CheckNum10

ADR r8, StringNum6 ;load string

printNum6 LDRB r0, [r8], #1 ;load counter into r0 if r0 equal to 0

CMP r0, #0 ;check null terminator

SWINE SWI\_WriteC ;print r0

BNE printNum6

B Exit

CheckNum10 CMP r3, #10

BNE Exit

ADR r8, StringNum10 ;load string

printNum10 LDRB r0, [r8], #1 ;load counter into r0 if r0 equal to 0

CMP r0, #0 ;check null terminator

SWINE SWI\_WriteC ;print r0

BNE printNum10

B Exit

String DCB "This is my 8string, and n99umbers", &0a, 0

StringNum1 DCB "1", 0

StringNum2 DCB "2", 0

StringNum3 DCB "3", 0

StringNum4 DCB "4", 0

StringNum5 DCB "5", 0

StringNum6 DCB "6", 0

StringNum7 DCB "7", 0

StringNum8 DCB "8", 0

StringNum9 DCB "9", 0

StringNum10 DCB "10", 0

StringNum11 DCB "11", 0

ALIGN

END



