ΜΙΚΡΟΕΠΕΞΕΡΓΑΣΤΕΣ

ΚΑΙ ΠΕΡΙΦΕΡΙΑΚΑ

1η ΕΡΓΑΣΙΑ

ΟΝ/ΕΠ : ΝΙΚΟΛΑΟΣ ΙΣΤΑΤΙΑΔΗΣ

ΑΕΜ : 9175

EMAIL: nikoista@ece.auth.gr

INTRODUCTION

Θα χρησιμοποιήσουμε Keil uvision 5 με όλα τα εργαλεία που μας παρουσιάστηκαν στο 1 εργαστήριο. Αρχικά δημιουργούμε το project μας στην uvision5 αφού πρώτα έχουμε κατεβάσει και εγκαταστήσει τα απαραίτητα αρχεία - compoments που βοηθούν στο compile-debug και γράψιμο του κώδικα μας. Στην συνέχεια θα υλοποιήσουμε κώδικα σε C άλλα και σε Arm Assembly, σύμφωνα με τις προδιαγραφές που μας έχουν δωθεί. Για το πρόγραμμα μας θα χρησιμοποιήσουμε το ST32F401RE επεξεργαστή του NUCLEO ARM .Ταυτόχρονα το simulation που θα κάνουμε είναι ένα virtual simultaion που προσφέρει το πρόγραμμα Keil uvision 5 για να κάνουμε debug το πρόγραμμα μας, μιας και δεν έχω αυτή την στιγμή την πλακέτα.

ΠΡΟΒΛΗΜΑΤΑ ΠΟΥ ΕΜΦΑΝΙΣΕ Ο DEBUGER

Καταρχάς εμφανιζόταν μονίμως ένα error65 : access violation at 0x40023800 όπου το εξαλείψαμε μέσο ενός KEIL_STM.ini αρχείο που μέσα έχει την εντολή MOV 0x40000000 , 0x400FFFFF READ,WRITE και το αποθήκευσα στον φάκελο του project μας. Επίσης μέσο του Debuger μπόρεσα να καταλάβω την δομή ενός κώδικά σε ARM Assembly και C αλλά και είδα τον τρόπο με τον οποίο το πρόγραμμα δεσμεύει τους registers ανάλογα με τον κώδικα που γράφεις. Έτσι για το πρόγραμμα της C είδα ότι ο r0-r1-r2-r3-r4-r14-r15-xPSR ενώ στο ARM Assembly είδα ότι ο r0-r1-r4-r6-r8-r9-r11 register δεσμέυοταν για τα στοιχεία μου και έβλεπα τις τιμές που είχαν μέσο του debuger. Μετά από πολύ ώρα και γραμμή - γραμμή debug συνειδητοποίησα ότι ο r0 και ο r1 είχαν τις θέσεις μνήμης των ορισμάτων που έστελνα στην routine stringCheck.asm .Εκεί ο pointer first_string δείχνει στον r0 και ο int size δείχνει στον r1.Κάνοντας τις απαραίτητες διεργασίες και στο τέλος γυρίζω στο C αρχείο τον r0 με'Y' ή'N' ,δηλαδή αν είναι το String παλινδρομικό ή όχι και τον r1 με 0 ή 1,αν όλα πήγαν καλά η είχα σφάλματα.

ΠΡΟΓΡΑΜΜΑ ΣΕ C

Χρησιμοποιούμε τα απαραίτητα headers αρχεία σε .h για το πρόγραμμα μας στην γλώσσα C και ενεργοποιούμε τον απαραίτητο compiler για να κάνουμε ένας σωστό build. Η βασική ρουτίνα int main(void) στο stringMain.c αρχείο μας θα δημιουργήσει και θα αρχικοποιήσει τον στατικό πίνακα (string) όπου θα ελεγχθεί στην συνέχεια από το Arm Assembly

ΚΩΔΙΚΑΣ ΣΕ C

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
extern int stringCheck(const
char *first_string,int size);

int main(void)
{
const char str[]="AaB BaA";
```

πρόγραμμα. Επίσης θα δημιουργήσουμε άλλες δύο μεταβλητές τύπου int , τις length και returnValue αλλά και την char isPalindromic όπου θα δόυμε και σε λίγο στον κώδικα. Στην ουσία δημιουργούμε ένα routine-function int stringCheck που έχει δύο ορίσματα , 1) την αρχή του string σε έναν pointer, 2) το μέγεθος του string. Καλούμε την συνάρτηση stringCheck με και δίνουμε τιμές στις μεταβλητές isPalindromic και returnValue σύμφωνα με τον έλεγχω που έγινε μέσα στην συνάρτηση.Τέλος εμφανίζουμε τα αποτελέσματα στην οθώνη και κλείνουμε με το while(1) ένας ατέρμον βρόγχος που τρέχει μέχρις ότου να υπάρξει ένα ξεκάθαρο BREAK STATEMENT .

```
char isPalindromic='\0';
int returnValue=0;
int length=0;

length=strlen(str);
isPalindromic=stringCheck(str,length-1);
returnValue=length;

printf("%s",&isPalindromic);
printf("%d",returnValue);
while(1);
}
```

ΠΡΟΓΡΑΜΜΑ ΣΕ ASM

Χρησιμοποιούμε τα απαραίτητα headers για το πρόγραμμα μας στην γλώσσα ARM Assembly και ενεργοποιούμε τον απαραίτητο compiler για να κάνουμε ένας σωστό build.Η ρουτίνα stringCheck παίρνει τα ορίσματα και εκτελέι έναν έλενχο για να δει αν το string είναι Παλινδρομικό. Άρα στην αρχή περνάμε τις διευθήνσεις-τιμές του r0 και r1 και ελέγχουμε ένα ένα τα στοιχεία του string αν είναι αντιστρόφος-καθρέπτης ίδια. Αν ναι τότε δώσε στο register r0 την τιμή Ύ΄και στον r1 την τιμή 1 ότι όλα πήγαν καλά.Αν όχι τότε δώσε στο register r0 την τιμή Ν΄και στον r1 την τιμή ότι υπήρξε σφάλμα.

ΠΡΟΒΛΗΜΑΤΑ ΚΑΙ TESTING

Μετά από πολλά testing στο Debuger και πολλές ώρες διάβοντας ARM Assembly decumentation .Εν κατακλείδι κατάλαβα ότι στον r0 στην C έχουμε βάλει το adress του 1 στοιχείου του Πίνακα str[] και στον r4 το adress του length οπότε αφού δεν θα τα χρειαστούμε στην συνέχεια του προγράμματος τα αλλάζω σύμφωνα με την ρουτίνα stringCheck και του δίνω τις τιμές που πρέπει να πάρουν μέσα απότ την stringCheck.

ΚΩΔΙΚΑΣ ΣΕ ARM ASM

```
      GLOBAL stringCheck
      AREA |.PalindromicString|,CODE,READONLY
      EXPORT stringCheck
StringCheck
      ADDS r6 , r1 , r0;
ADDS r1 , r1 , #1;
LDRB r9 , [r0] ;
LDRB r11 , [r6] ;
       CMP r9 , #'\0' ;
CMP r11 , #'\0' ;
BNE Loop ;
SUBS r0 , r0 , r0;
ADD      r0 , #'Y';                    SUBS
r4, r4 , r4;
ADD r4 , #1;
               BX lr;
Loop
LDRB r9 , [r0] ;
LDRB r11 , [r6] ;
CMP r9 , r11 ;
BEQ Palindromic;
SUBS r0 , r0 , r0 ;
ADD      r0 , #'N' ;
SUBS r4 , r4 , r4 ;
       BX lr;
Palindromic
ADD r0 ,  #1 ;                         SUB r6
,  #1 ;
ADDS r8 , r8 , #1 ;
CMP r8 , r1 ;
BNE Loop ;
SUBS r0 , r0 , r0 ;
ADD      r0 , #'Y' ;
SUBS r4 , r4, r4;
ADD r4 , #1;
BX lr;
END
```