ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ

ΤΜΗΜΑ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΙΣΤΩΝ

ΣΧΕΔΙΑΣΜΟΣ ΕΝΣΩΜΑΤΩΜΕΝΩΝ ΣΥΣΤΗΜΑΤΩΝ – 7ο ΕΞΑΜΗΝΟ

Μαρκέτος Νικόδημος – ΑΜ:03117095

Τζομάκα Αφροδίτη – ΑΜ: 03117107

3η ΕΡΓΑΣΤΗΡΙΑΚΗ ΑΣΚΗΣΗ

Ερώτημα 1ο: Μετατροπή εισόδου από τερματικό

Αρχικά, η επικοινωνία με το terminal υλοποιήθηκε μέσω system calls.

Διαβάζουμε λοιπόν (read) από το standard input και αποθηκεύουμε τα δεδομένα που θα εισάγει ο χρήστης σε έναν buffer (input_str). Ελέγχουμε αν πατήθηκε μόνο ένας χαρακτήρας και αυτός είναι q ή Q ώστε να κάνουμε exit. Σε αντίθετη περίπτωση καλούμε την συνάρτηση που ουσιαστικά εκτελεί την μετατροπή. Σε αυτήν αρχικά σώζουμε στην στοίβα τους καταχωρητές που επηρεάζονται (r5) καθώς και τον link register τον οποίο αργότερα θα κάνουμε pop στον program counter για να επιστρέψουμε ορθά στην ροή του main προγράμματος. Στην συνέχεια επιτελείται η λειτουργία που ζητείται από την άσκηση δηλαδή για τα δεδομένα που είναι αριθμοί 0-4 προσθέτουμε 5, για τα δεδομένα που είναι αριθμοί 5-9 αφαιρούμε 5 και για τα δεδομένα που είναι χαρακτήρες εναλλάσσουμε τα κεφαλαία σε μικρά και το αντίστροφο.

Εκτυπώνουμε στην οθόνη (write) ως output μόνο 32 χαρακτήρες (αγνοώντας τους υπόλοιπους όπως ζητείται). Αν δόθηκαν περισσότεροι από 32 χαρακτήρες χρειαζόμαστε ένα newline για το format του output.

*Αρχείο άσκησης ex1s.s

Ερώτημα 2ο: Επικοινωνία των guest και host μηχανημάτων μέσω σειριακής θύρας.

Εκτελούμε τις οδηγίες της άσκησης και πλέον έχουμε συνδεδεμένο τον host με μια σειριακή θύρα (dev/pts/<number>) η οποία δίνεται κάθε φορά που εκκινούμε το qemu μέσω του ορίσματος –serial pty ενώ ο guest με την σειριακή θύρα /dev/ttyAMA0.

- host_improved.c

Στον host δίνεται ως όρισμα η σειριακή θύρα η οποία ανοίγεται για την επικοινωνία. Ελέγχουμε ότι όλα πήγαν καλά και το ανοιγμένο αρχείο είναι tty (istty()) και ξεκινάμε το configuration. Έχει οριστεί μια δομή τύπου termios, η config, και σε αυτήν ορίζουμε τα αντίστοιχα flags. Γενικά επειδή δεν έχουμε ειδικούς περιορισμούς τα μόνα flags που πειράζουμε είναι τα εξής (εξηγούνται κ αναλυτικότερα στα σχόλια της assembly του guest που ακολουθεί):

```
memset(&config, 0, sizeof(config));

config.c_iflag = 0;

config.c_oflag = 0;

config.c_lflag = 0;

config.c_cc[VMIN] = 1;  //One input byte is enough to return from read()

config.c_cc[VTIME] = 0; //Inter-character timer off

config.c_cflag = B9600 | CS8 | CLOCAL | CREAD;
//Baud 9600, 8-bit data, CLOCAL and CREAD to ensure that

//(CLOCAL) the program does not become the 'owner' of the port subject to sporatic
job control and hangup signals,

//(CREAD) and also that the serial interface driver will read incoming data bytes
```

Στην συνέχεια εκτελούμε την tcsetattr() ώστε να περαστούν οι αλλαγές και tcflush() για να αδειάσουμε το pseudoterminal του termios από πιθανά «σκουπίδια».

Διαβάζουμε την συμβολοσειρά από το standard input και τα γράφουμε στην σειριακή θύρα του host. Έπειτα καλείται η συνάρτηση tcdrain() ώστε να περιμένει το πρόγραμμα την αποστολή των δεδομένων μέσω του tty πριν προχωρήσει.

Τέλος, διαβάζονται ένα – ένα ο χαρακτήρας και η συχνότητα εμφάνισής του και εκτυπώνονται στην οθόνη.

Κλείνουμε τον file descriptor της σειριακής θύρας και επιστρέφουμε.

- guest_improved.s

Γενικά και εδώ χρησιμοποιούνται system calls για τα open, read, write, exit ωστόσο κάνουμε και extern τις συναρτήσεις που αφορούν το configuration του termios (tcsetattr(), tcflush()).

Από την μεριά του guest ανοίγουμε την σειριακή θύρα /dev/ttyAMA0 με flags O_RDWR|O_NOCTTY|O_NONBLOCK το οποίο έχει δεκαεξαδική τιμή 0x902 και αποθηκεύουμε τον file descriptor σε έναν καταχωρητή. Εκτελούμε το configuration σύμφωνα με τα αντίστοιχα options:

```
options : .word 0x00000000 /* c_iflag */

.word 0x00000000 /* c_oflag */

.word 0x000008bd /* c_cflag - Control options*/

//Perfoming AND operation between the following flags we get 0x08bd in hex.

//Baud = B9600 -> 0000015

//8 bits data = CS8 -> 0000060

//CREAD = Enable receiver -> 0000200

//CLOCAL = Ignore modem status lines-> 0004000

/*The c_cflag member contains two options that should always be enabled, CLOCAL
and CREAD.

These will ensure that your program does not become the 'owner' of the port
subject to sporatic job control and hangup signals,

and also that the serial interface driver will read incoming data bytes.*/

.word 0x00000000 /* c_lflag */

.byte 0x00 /* c_line */

.word 0x00000000 /* c_cc[0-3] */

.word 0x00010000 /* c_cc[4-7] */

//From termios.h -> byte 6 is for VMIN (little endian)

//VMIN: Minimum number of characters to read

.word 0x00000000 /* c_cc[8-11] */

.word 0x00000000 /* c_cc[12-15] */

.word 0x00000000 /* c_cc[16-19] */

.word 0x00000000 /* c_cc[20-23] */

.word 0x00000000 /* c_cc[24-27] */

.word 0x00000000 /* c_cc[28-31] */

.byte 0x00 /* padding */

.hword 0x0000 /* padding */

.word 0x0000000 /* c_ispeed */

.word 0x0000000 /* c_ospeed */
```

Οι τιμές αυτές προέκυψαν αφού συμβουλευτήκαμε και το αντίστοιχο αρχείο termios.h.

Συνεχίζοντας στην ροή του προγράμματος, αφού εκτελεστεί το απαραίτητο tcflush που αναφέρθηκε και από την πλευρά του host, περνάμε στην κυρίως λειτουργία του προγράμματος.

Χρησιμοποιούμε έναν πίνακα (chars) 256 θέσεων, μία για κάθε ascii χαρακτήρα. Διαβάζουμε ένα – ένα τα bytes από την σειριακή θύρα και για κάθε ένα αν:

1. Είναι χαρακτήρας ascii,
2. δεν είναι κενό και
3. δεν είναι newline

πηγαίνουμε στο index που καταδεικνύει η δεκαδική τιμή του ascii χαρακτήρα που πατήθηκε και αυξάνεται η συχνότητά εμφάνισής του κατά 1.

Μόλις πατηθεί το newline ξεκινάει η διαδικασία εύρεσης του μεγίστου σε αυτόν τον πίνακα (chars). Το μέγιστο στοιχείο αντιστοιχεί στην μέγιστη συχνότητα εμφάνισης ενώ το index του στοιχείου αυτού στον πιο συχνά εμφανιζόμενο χαρακτήρα. Αποθηκεύονται σε δύο δεσμευμένες θέσεις μνήμης (char, freq) και στέλνονται στον host ένας – ένας (2 write).

Κλείνουμε το αρχείο (close) και επιστρέφουμε (exit).

Ερώτημα 3ο: Σύνδεση κώδικα C με κώδικα assembly του επεξεργαστή ARM.

Σε ένα κοινό αρχείο, str_fun.s, υλοποιούμε τις συναρτήσεις strlen, strcpy, strcmp και strcat. Ο κώδικας είναι απλός και αντιστοιχεί άμεσα στην λειτουργία που εκτελούν οι συναρτήσεις αυτές και στην c.

Εκτελούμε το αρχείο string_manipulation.c για κάθε rand_input που μας έχει δοθεί και αποθηκεύουμε τα out αρχεία.

Στην συνέχεια, μέσα στο qemu, εκτελούμε το αντίστοιχο makefile και παράγουμε ξανά out αρχεία για κάθε input file, αυτή την φορά όμως έχοντας κάνει link το αρχείο string_manipulation.c με το str_fun.s προσθέτοντας στο πρώτο το εξής τμήμα κώδικα:

```
extern size_t strlen(char *s);
extern char * strcpy(char *dest, char *src);
extern char * strcat(char *dest, char *src);
extern int strcmp(char *s1, char *s2);
```

προκειμένου πλέον οι συναρτήσεις strlen, strcpy, strcmp και strcat να είναι αυτές που υλοποιήσαμε εμείς.

Εκτελώντας την εντολή diff ανάμεσα στα πρώτα outputs που παρήχθησαν από τον original κώδικα και σε αυτά που παρήχθησαν από τον κώδικα που περιέχει τις δικές μας υλοποιήσεις βλέπουμε ότι πήραμε τα ίδια, σωστά αποτελέσματα.