ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ

ΤΜΗΜΑ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΙΣΤΩΝ

ΕΡΓΑΣΤΗΡΙΟ ΜΙΚΡΟΥΠΟΛΟΓΙΣΤΩΝ – 7ο ΕΞΑΜΗΝΟ

Δημητρίου Αγγελική – ΑΜ: 03117106

Τζομάκα Αφροδίτη – ΑΜ: 03117107

Ομάδα Β8

5η Εργαστηριακή Άσκηση

Η αναφορά αυτή περιλαμβάνει τους κώδικες σε c που γράφηκαν για τα δύο ζητούμενα της άσκησης καθώς και την assembly εκδοχή τους που προέκυψε αυτόματα από τον debugger. Η assembly συνοδεύεται με αναλυτικά σχόλια πλάι από κάθε εντολή τα οποία εξηγούν τη ροή των προγραμμάτων.

ΖΗΤΗΜΑ 5.1

```c
#define GPIO_SWs    0x80001400      //RISCV little endian
#define GPIO_LEDs   0x80001404
#define GPIO_INOUT  0x80001408      //2LSBytes for LEDs, 2MSBytes for Switches
                                    //Input = 0, Output = 1

#define READ_GPIO(dir) (*(volatile unsigned *)dir)
#define WRITE_GPIO(dir, value) {(*(volatile unsigned *)dir) = (value);}

int main (void)
{
    int io_def = 0xFFFF; //Input from switches, Output at LEDs
    int sw_val, tmp;

    WRITE_GPIO(GPIO_INOUT, io_def);

    while(1)
    {
        sw_val = READ_GPIO(GPIO_SWs);
        sw_val = sw_val >> 16;  //Shift into lower 16bits
        tmp = sw_val & 0x0F;    //Mask the 4 LSBs
        sw_val = sw_val >> 12;  //To bring 4 MSBs to 4 LSBs
        sw_val += tmp;
        if (sw_val >= 0x10)     //Checking for overflow
        {
            sw_val = 0x10;
        }
        WRITE_GPIO(GPIO_LEDs, sw_val);
    }
```

		Πρώτα εκτελείται το write που θέτει εισόδους και εξόδους.
0x00000090: 37 17 00 80	**lui a4,0x80001**	***Αρχικά, τοποθετώντας τον κατάλληλο αριθμό στα πιο σημαντικά bits του a4, ο ίδιος έχει πλέον τον αριθμό 0x80001000 (θα χρησιμοποιηθεί ως βάση για να γράφουμε ή διαβάζουμε από τις διευθύνσεις που αφορούν την πρόσβαση σε led και switches).***
0x00000094: c1 67 **0x00000096: fd 17**	**lui a5,0x10** **addi a5,a5,-1**	***Με την ίδια λογική στον a5 τοποθετείται ο αριθμός 0x00010000 που με την αφαίρεση του 1 γίνεται 0x0000ffff που θέλουμε να γράψουμε στην 80001408 που ορίζει είσοδο, έξοδο***
0x00000098: 23 24 f7 40	**sw a5,1032(a4) # 0x80001408**	***Η εγγραφή του a5 γίνεται εν τέλει στη σωστή διεύθυνση με προσθήκη offset 1032 ή 0x408.***
0x0000009c: 29 a0	**j 0xa6 <main+22>**	***Κάνουμε jump στο κομμάτι που διαβάζει από switches.***
0x0000009e: 37 17 00 80 **0x000000a2: 23 22 f7 40**	**lui a4,0x80001** **sw a5,1028(a4) # 0x80001404**	***Τοποθετούμε στον a4 τη βάση για τη διεύθυνση όπου θα γράψουμε με offset 1028 ή 0x404 γράφουμε στη διεύθυνση των leds την τιμή που είναι αποθηκευμένη στον a5.***
0x000000a6: b7 17 00 80 **0x000000aa: 03 a7 07 40**	**lui a5,0x80001** **lw a4,1024(a5) # 0x80001400**	***Τοποθετούμε τώρα στον a5 τη βάση για τη διεύθυνση από όπου θα διαβάσουμε με offset 1024 ή 0x400 διαβάζουμε από τη διεύθυνση των switches την τιμή τους και την τοποθετούμε στον a4.***
0x000000ae: 93 57 07 41	**srli a5,a4,0x10**	***Ολίσθηση κατά 16 bits της τιμής που διαβάστηκε στην είσοδο, τα 4 lsb της εισόδου έρχονται στα lsb και η τιμή τοποθετείται στον a5.***
0x000000b2: bd 8b **0x000000b4: 71 87**	**andi a5,a5,15** **srli a4,a4,0x1c**	***Μάσκα στα 4 lsb.*** ***Ολίσθηση κατά 12 bits της τιμής που διαβάστηκε στα 4 msb της εισόδου για να έρθει στα 4 lsb του a4.***
0x000000b6: ba 97	**add a5,a5,a4**	***Πρόσθεση των 2 τιμών και τοποθέτηση στον a5.***
0x000000b8: 3d 47 **0x000000ba: e3 52 f7 fe**	**li a4,15** **bgeu a4,a5,0x9e <main+14>**	***Τοποθέτηση του 15 στον a4 και σύγκριση του αθροίσματος με αυτόν για εύρεση υπερχείλισης. Όταν δεν υπάρχει υπερχείλιση (a5<16) απλά γίνεται μετάβαση στο κομμάτι που γράφει στα led***
0x000000be: c1 47 **0x000000c0: f9 bf**	**li a5,16** **j 0x9e <main+14>**	***Όταν υπάρχει υπερχείλιση, πρώτα φορτώνουμε στον a5 το 16 που ανάβει το πέμπτο led (μόνο αυτό) και δείχνουμε την τιμή αυτή στα led. Προφανώς προκύπτει συνεχής λειτουργία του προγράμματος.***

ZHTHMA 4.2

```c
#define GPIO_SWs 0x80001400 //RISCV little endian
#define GPIO_LEDs 0x80001404
#define GPIO_INOUT 0x80001408 //2LSBytes for LEDs, 2MSBytes for Switches
                              //Input = 0, Output = 1

#define DELAY 0x300000

#define READ_GPIO(dir) (*(volatile unsigned *)dir)
#define WRITE_GPIO(dir, value)
    {
        (*(volatile unsigned *)dir) = (value);
    }

int main(void)
{
    int io_def = 0xFFFF; //Input from switches, Output at LEDs
    int flag = 0xFFFF;   //Flag for blinking
    unsigned int sw_val, tmp, msb, out_val, cnt, i, del, neg;

    WRITE_GPIO(GPIO_INOUT, io_def);

    sw_val = READ_GPIO(GPIO_SWs); //Read from switches

    while (1)
    {
        msb = sw_val & 0x80000000; //mask msb of input
        neg = ~(sw_val >> 16); //Shift into lower 16bits and complement
        tmp = neg;

        //Count 1s in negation
        do
        {
            if (tmp & 0x01 == 1)
                cnt++;
            tmp = tmp >> 1;

        } while (tmp != 0xFFFF);
        cnt *= 2;
        //Blinking
        do
        {
            out_val = neg & flag;
            WRITE_GPIO(GPIO_LEDs, out_val); //Display
            del = DELAY;
            i = 0;
            while (i < del)
                i++;
            flag = flag ^ 0xFFFF;
            cnt--;
        } while (cnt != 0);

        WRITE_GPIO(GPIO_LEDs, 0x00);

        //sw_val = READ_GPIO(GPIO_SWs);

        do
        {
            sw_val = READ_GPIO(GPIO_SWs);
        } while ((sw_val & 0x80000000) == msb);
    }
```

0x00000090: b7 17 00 80	**lui**	**a5,0x80001**	***Όπως στην προηγούμενη άσκηση ορίζουμε είσοδο-έξοδο με αυτές τις 4 εντολές.***
0x00000094: 41 66	**lui**	**a2,0x10**	
0x00000096: 7d 16	**addi**	**a2,a2,-1**	
0x00000098: 23 a4 c7 40	**sw**	**a2,1032(a5) # 0x80001408**	
0x0000009c: 83 a7 07 40	**lw**	**a5,1024(a5)**	***Αποθηκεύεται στον a5 το περιεχόμενο των switches (θέση 0x80001 + 0x400 offset)***
0x000000a0: a1 a8	**j**	**0xf8 <main+104>**	***Πηγαίνει στις αρχικοποιήσεις***
			Κομμάτι προγράμματος που μετράμε τους άσσους στην άρνηση:
0x000000a2: 85 83	**srli**	**a5,a5,0x1**	***Ολίσθηση του a5(tmp) μια μονάδα δεξιά (στα MSB τοποθετούνται μηδενικά - unsigned).***
0x000000a4: 41 67	**lui**	**a4,0x10**	
0x000000a6: 7d 17	**addi**	**a4,a4,-1**	***Στον a4 έχουμε 0xffff (0x10000 - 0x1).***
0x000000a8: 63 87 e7 00	**beq**	**a5,a4,0xb6 <main+38>**	***Όσο ο a5 δεν είναι 0xffff (έχει προηγηθεί άρνηση της αρχικής τιμής της εισόδου),***
0x000000ac: 13 f7 17 00	**andi**	**a4,a5,1**	***βάζουμε μάσκα στο τελευταίο***
0x000000b0: 6d db	**beqz**	**a4,0xa2 <main+18>**	***bit του a5, τον μεταφέρουμε στον a4 και αν είναι 0 (δε βρήκαμε άσσο) θα συνεχίσουμε το shift.***
0x000000b2: 85 06	**addi**	**a3,a3,1**	***Αλλιώς αυξάνουμε των μετρητή***
0x000000b4: fd b7	**j**	**0xa2 <main+18>**	***των άσσων και συνεχίζουμε το shift.***
0x000000b6: 86 06	**slli**	**a3,a3,0x1**	***Πολλαπλασιάζουμε το άθροισμα των άσσων γιατί οι διαφορετικές καταστάσεις που θα βρεθούν τα leds είναι διπλάσιες από τα αναβοσβήματα.***
0x000000b8: 31 a0	**j**	**0xc4 <main+52>**	***Πηγαίνει στο κομμάτι που φτιάχνει την τιμή εξόδου.***
0x000000ba: c1 67	**lui**	**a5,0x10**	***Στον a5 το 0xffff.***
0x000000bc: fd 17	**addi**	**a5,a5,-1**	
0x000000be: 3d 8e	**xor**	**a2,a2,a5**	***Στον a2 μπαίνει το xor του ίδιου με το 0xffff δηλαδή παίζει το ρόλο του flag που δείχνει αν τα leds ανάβουν ή σβήνουν βάζοντας την κατάλληλη μάσκα σε αυτά.***
0x000000c0: fd 16	**addi**	**a3,a3,-1**	***Ο a3 έχει πόσες φορές θα***
0x000000c2: 91 ce	**beqz**	**a3,0xde <main+78>**	***ανάψουν ή σβήσουν τα φώτα, άρα κάθε φορά μειώνεται κατά 1.***
			Όταν μηδενιστεί πάμε στο επόμενο βήμα.
0x000000c4: b3 77 b6 00	**and**	**a5,a2,a1**	***Ο a1 έχει την άρνηση της εισόδου, του βάζουμε τη σωστή μάσκα για άναμμα ή σβήσιμο.***

0x000000c8: 37 17 00 80 0x000000cc: 23 22 f7 40	lui a4,0x80001 sw a5,1028(a4) # 0x80001404	*Δείχνουμε το αποτέλεσμα στα leds όπως στην προηγούμενη άσκηση.*
0x000000d0: 81 47 0x000000d2: 37 07 30 00	li a5,0 lui a4,0x300	*Εισάγουμε καθυστέρηση:* *Στον a5 η αρχική τιμή μετρητή «δευτερολέπτων», στον a4 η επιθυμητή καθυστέρηση.*
0x000000d6: e3 f2 e7 fe 0x000000da: 85 07	bgeu a5,a4,0xba <main+42> addi a5,a5,1	*Όσο δεν είναι ίσα απλά αυξάνουμε τον a5 κατά 1, αλλιώς φτιάχνουμε σωστά το flag.*
0x000000dc: dd bf	j 0xd2 <main+66>	
0x000000de: b7 17 00 80 0x000000e2: 23 a2 07 40	lui a5,0x80001 sw zero,1028(a5) # 0x80001404	*Μηδενίζουμε τα leds.*
0x000000e6: b7 17 00 80 0x000000ea: 83 a7 07 40	lui a5,0x80001 lw a5,1024(a5) # 0x80001400	*Με αυτές τις εντολές διαβάζονται τα switches και η καινούρια τιμή εισόδου τοποθετείται στον a5*
0x000000ee: 37 07 00 80 0x000000f2: 7d 8f	lui a4,0x80000 and a4,a4,a5	*Τοποθετείται μάσκα στο MSB της καινούριας τιμής των διακοπτών*
0x000000f4: e3 09 a7 fe	beq a4,a0,0xe6 <main+86>	*Η προηγούμενη τιμή συγκρίνεται με το MSB της αρχικής εισόδου που αποθηκεύσαμε κατά την αρχικοποίηση. Όσο είναι ίδια ξαναδιαβάζουμε*
0x000000f8: 37 05 00 80 0x000000fc: 7d 8d	lui a0,0x80000 and a0,a0,a5	*Αρχικοποιήσεις:* *Για να κρατήσουμε το msb βάζουμε κατάλληλη μάσκα στην είσοδο (a5) και το αποτέλεσμα μένει στον a0*
0x000000fe: c1 83 0x00000100: 93 c5 f7 ff	srli a5,a5,0x10 not a1,a5	*Στον a1 μπαίνει η άρνηση της εισόδου αφού γίνει shift 16 bit (η τιμή των switches αποθηκεύεται αρχικά στα 16 ανώτερα bit)*
0x00000104: ae 87	mv a5,a1	*Ένα αντίγραφο κρατιέται στον a5 (tmp)*
0x00000106: 5d b7	j 0xac <main+28>	*Επιστρέφει στο μέτρημα άσσων*