Πρώτο εργαστήριο RISC V

Εργαστήριο Μικροϋπολογιστών

Παγώνης Γεώργιος Α.Μ.: 03117030

1η Άσκηση : Add_4_bit_numbers

Κώδικας:

```
#define GPIO_SWs 0x80001400
#define GPIO_LEDs 0x80001404
#define GPIO_INOUT 0x80001408
#define READ_GPIO(dir) (*(volatile unsigned *)dir)
#define WRITE_GPIO(dir, value)
  {
    (*(volatile unsigned *)dir) = (value);
  }
int main(void)
{
  int En_Value = 0xFFFF;
  WRITE_GPIO(GPIO_INOUT, En_Value);
  unsigned int first_number, second_number;
  unsigned int switches, output;
  while (1)
  {
    switches = READ_GPIO(GPIO_SWs); // read value on switches
    first_number = switches >> 16;
    second_number = switches >> 28;
    first_number = first_number & 0x000F;
    second_number = second_number & 0x000F;
    output = first_number + second_number;
    if (output > 0x000F)
    {
      output = 0x0010;
      WRITE_GPIO(GPIO_LEDs, output);
    }
```

```
    else
    {
      WRITE_GPIO(GPIO_LEDs, output);
    }
 }
return (0);
}
```

Περιγραφή της άσκησης:

- Διαβάζουμε την είσοδο από την θέση 0x80001400 . Όμως επειδή τα δεδομένα είναι στα 16 ποιο σημαντικά bits τα μεταφέρουμε με shift left >> 16 και για να πάρουμε τα 4 πρώτα bits από την είσοδο τα απομονώνουμε με την add 0x000F . Το ίδιο κάνουμε και για να πάρουμε τα τελευταία 4 bits της εισόδου και απονώνουμε με τον ίδιο τρόπο . Τα προσθέτουμε , και χρησιμοποιούμε ένα if statement όπου άμα το άθροισμα είναι μεγαλύτερο του 0x000F και έχουμε υπερχείλιση τότε απλά ανάβουμε το 5 bit της εξόδου , αλλιώς τοποθετούμε στην έξοδο το αποτέλεσμα του αθροίσματος.

Assembly:

```
lui a4,0x80001 ;φορτώνει στον καταχωρητή α4 την θεση 0x80001
lui a5,0x10 ; φορτώνει τον α5 με 0X10
addi    a5,a5,-1 ;αφαιρεί ένα από το α5 για να δημιουργήσει το 0x0f
sw  a5,1032(a4) # 0x80001408; το αποθηκεύει για να κάνει έλεγχο εισόδου με 1->έξοδος
j   0xa6 <main+22> ; ξεκινάει το κύριο πρόγραμμα με την εντολή while
lui a4,0x80001
sw  a5,1028(a4) # 0x80001404 ; αποθήκευσε το output
lui a5,0x80001
lw  a5,1024(a5) # 0x80001400; το α5 έχει την είσοδο σαν ένα 32 bit
srli    a4,a5,0x10 ; βάζουμε τα πρώτά 4 bits στο α4
srli    a5,a5,0x1c ; βάζουμε τα 4 τελευταία bits στο α5
andi    a4,a4,15 ; κάνε mask τα στοιχεία
add a5,a5,a4; τα προσθέτουμε και το συνολικό αποτέλεσμα το βάζουμε στο α5
li  a4,15 ; βάζουμε στο α4 το 0x0F για κάνουμε την σύγκριση στο if
bgeu    a4,a5,0x9e <main+14> ; άμα είναι greater or equal α4>=α5 τότε πήγαινε στην θέση
```

```
                          ; αυτή η θέση είναι για να περάσει  την γραμμή li  a4,16 που δεν
                            ; την θέλουμε απλά βάζουμε το αποτέλεσμα στην έξοδο
lui a5,0x80001
li  a4,16 ; φορτώνει το α4 με το 0x010;
sw  a4,1028(a5) # 0x80001404 ; τοποθετεί την έξοδο το αποτέλεσμα που βρίσκεται στον α4 άμα
έχουμε κάνει το if η όχι
j   0xa6 <main+22> ; γυρίζει στην αρχή
```

2η Άσκηση : Show_the_negative

Κώδικας:

```
// memory-mapped I/O addresses
#define GPIO_SWs 0x80001400
#define GPIO_LEDs 0x80001404
#define GPIO_INOUT 0x80001408
#define READ_GPIO(dir) (*(volatile unsigned *)dir)
#define WRITE_GPIO(dir, value)             \
  {                                        \
    (*(volatile unsigned *)dir) = (value); \
  }
int main(void)
{
  int En_Value = 0xFFFF;
  WRITE_GPIO(GPIO_INOUT, En_Value);
  unsigned int first_number, switches, output;
  unsigned int prev_temp;
  unsigned int second = 0x0000;
  //change the value in output line
  //test values:0b0101010101010101=21845=0x5555 output:0xAAAA
  //test values:0b0000000000101001=41=0x29 output :0xffd6
```

```
//best practice to change the value in the first number line
//by changing the switches value
//0x55550000 = 1431633920 output:0xAAAA
//0x00290000=2686976 output :0xffd6
switches = READ_GPIO(GPIO_SWs); // read value on switches
first_number = switches >> 16;
output = first_number & 0xFFFF ^ 0xFFFF;
prev_temp = first_number >> 15 & 0x0001;
int i;
for (i = 0; i < 8; i++)
{
  WRITE_GPIO(GPIO_LEDs, output);
  //delay
  WRITE_GPIO(GPIO_LEDs, second);
  //delay
}
while (1)
{
  //change the value in output line
  //test values:0b0101010101010101=21845=0x5555 output:0xAAAA
  //test values:0b000000000101001=41=0x29 output :0xffd6
  //best practice to change the value in the first number line
//by changing the switches value
//0x55550000 = 1431633920 output:0xAAAA (this has 0 in the msb)
//0x00290000=2686976 output :0xffd6
//to check what we need to change the msb to change the values
//0xd5550000= 3579117568 (this has 1 in the msb)
  unsigned int current_temp;
  switches = READ_GPIO(GPIO_SWs); // read value on switches
  first_number = switches >> 16;
```

```
    output = first_number & 0xFFFF ^ 0xFFFF;
    current_temp = first_number >> 15 & 0x0001;
    if (prev_temp != current_temp)
    {
      int i;
      for (i = 0; i < 8; i++)
      {
        WRITE_GPIO(GPIO_LEDs, output);
        //delay
        WRITE_GPIO(GPIO_LEDs, second);
        //delay
      }
      prev_temp = current_temp;
    }
  }
  return (0);
}
```

Περιγραφή της άσκησης:

Διαβάζουμε την είσοδο που είναι στα τελευταία 16 bit των switches τα μεταφέρουμε στα 16 πρώτα του first number , τα απονώνουμε και μετά κάνουμε xor για να πάρουμε το ζητούμενο αποτέλεσμα . Τέλος το τυπώνουμε για να αναβοσβήνει για 8 φορές.

Assemby:

(δεν εμφανίζονται τα masks πουθενά που τα τοποθετήσαμε στον c κώδικα για να σιγουρευτούμε για το αποτέλεσμα όταν κάνουμε shift right , εδώ `srli` γεμίζει από μόνο του τα 16 πρώτα bits με 0 .)

```
lui a4,0x80001
lui a5,0x10
addi    a3,a5,-1 # 0xffff ; a3=0xffff En_value=0xffff;
sw  a3,1032(a4) # 0x80001408 ; Αρχικοποιεί την έξοδο| WRITE_GPIO(GPIO_INOUT, En_Value);

lw  a4,1024(a4) ; a4=switches | παίρνει την τιμή από τον READ_GPIO
```

```
srli      a5,a4,0x10 ; first_number = switches >> 16;| a5 έχει το αριθμό των switches μετατοπισμένο κατά 16 θέσεις
not a5,a5 ; (θα έπρεπε να κάνει xor?) είναι το ίδιο πράγμα με την not
and a5,a5,a3 ; a5 = output  υπολογίζει το output
srli    a4,a4,0x1f ;shift right the a4= prev_temp κρατάει το msb ψηφίο
li  a3,0 ; a3=i |είναι το i το οποίο γίνεται addi    a3,a3,1 στο τέλος κάθε επανάληψης
j   0xc0 <main+48>
lui a2,0x80001 ; load upper imidiate για να μπορει να το χρησιμοποίσει σαν βάση για το sw
sw  a5,1028(a2) # 0x80001404 ; δείξε το output που είναι το a5 στην έξοδο
sw  zero,1028(a2) ; δείξε το 0 στην έξοδο για να κάνει το αναβόσβημα (δεν έχει κάτι ανάμεσα γιατί δεν έχουμε delay)
addi    a3,a3,1 ; αύξησε το i
li  a2,7 ; φόρτωσε την τιμή 7 στον a2 για να γίνει η σύγκριση
bge a2,a3,0xb2 <main+34> ; κάνε την σύγκριση και άμα είναι ίσα skipαρε το jump την επόμενη γραμμή
 j   0xdc <main+76> ; κανει μια τελευταία επανάληψη στο  blink των leds
 lui a2,0x80001
 sw  a5,1028(a2) # 0x80001404
sw  zero,1028(a2)
 addi    a3,a3,1 ; i++
 li  a2,7 ; a2=7
 bge a2,a3,0xc8 <main+56> ; μέχρι εδώ
mv  a3,a4 ; a3= prev_temp για να κραταει το msb
 lui a5,0x80001
lw  a4,1024(a5) # 0x80001400 ; φόρτωσε στο a4 το καινούργιο switches από READ_GPIO
srli    a5,a4,0x10 ; a5 = first_number >>16 για να έρθει στην σωστή θέση
not a5,a5 ; υπολόγισε την xor
slli    a5,a5,0x10; λογικά αύτο γίνεται για να γεμίσει τα 16 πρώτα bits με 0
srli    a5,a5,0x10
srli    a4,a4,0x1f; a4= msb από τα switches
beq a4,a3,0xde <main+78> ; άμα είναι ίσα τα prev_temp και current_temp που είναι τα 2 αποθηκευμένα msb τότε μην  κάνεις το blink
li  a3,0      ; αλλιώς αρχικοποίησε το i και κάνε τη
 j   0xd6 <main+70>
```

}