Δεύτερο εργαστήριο RISC V

Εργαστήριο Μικροϋπολογιστών

Παγώνης Γεώργιος Α.Μ.: 03117030

Κυριακόπουλος Ιωάννης Α.Μ. :03117440

1η Άσκηση: Abs_array

Κώδικας:

```
.globl main
.equ N, 10
.data
A: .word 0,1,2,7,-8,4,5,-12,11,-2 //0x2160 0x28
B: .word 0,1,2,7,-8,4,5,12,-11,-2 //0x2188 0x28
.bss
C: .space 4*N //0x21C0 0x28
//this numbers may change between different exe but
//we can check the position of the tables by oppening the
//memory in 0x8000140c +0x8
//add break point to sw a3,0(t2) to see the table changing
//by running and open the memory in 0x21C0 0x28
.text
main:
//display the position of all the tables in 0x8000140c +0x8  to know where to open the
memory
    la t0, A
    lui a5,0x80001
    sw t0,1036(a5)
    la t0,B
    sw t0,1040(a5)
    la t0,C
    sw t0,1044(a5)


nice:
    li a0,0 //init the counter to run this N times
    li a1,4*(N) //top end
    la t0, A //take the position of all the tables in t0,t1,t2
    la t1, B
    la t2, C
```

```
loop_make_c:
    lw a3,0x0(t0) //take the element of A that t0 points to =>start from 0
    lw a4,0x24(t1) // take the element of B that t1 points to => start from the 24 the N-
                    //1 element
    add a3,a3,a4 //add the 2 elements of the tables
    blt a3,zero,abs_C // if the result is below zero make it positive

return_form_abs_C:
    sw a3,0(t2) //store the result in the position the t2 points to
    add t0,t0,4 // make the pointer points to the next element of the array (we are going f
                    //orward)
    add t1, t1,-4 //make the pointer points to the previous element of the array(we are goi
              //ng backword)
    add t2,t2,4 //make the pointer of the result points to the next element to fill up
    add a0,a0,4 // add the counter to run N times the for loop
    bne a0,a1,loop_make_c // if we haven't reach the top end repeat
j nice


abs_C:
    not a3,a3 // take the supplement of 1 of the result
    dd a3,a3,1 // take the supplement of 2 of the result
    j return_form_abs_C
.end
```

Περιγραφή της άσκησης:

- Στην άσκηση αποθηκεύουμε τους πίνακες σε μια μεταβλητή A,B με την εντολή word και δεσμεύουμε και έναν χώρο για το αποτέλεσμα C με την εντολή space. Όμως επειδή ο compliler δεν τοποθετεί τα κομμάτια αυτά της μνήμης στο ίδιο σημείο (εξαρτάται και από τον υπόλοιπο κώδικα) , κάνουμε print την θέση των πινάκων σε μια γνωστή θέση μνήμης , για να μπορούμε να βλέπουμε το αποτέλεσμα του C . Στην συνέχεια ξεκινάμε να κάνουμε τις πράξεις έχοντας 2 pointers t0 , t1 που δείχνουν στην αρχή του A και στο τέλος του B αντίστοιχα. Παίρνουμε τις τιμές που δείχνουν οι 2 αυτοί pointers και τις προσθέτουμε . Κοιτάμε αν το αποτέλεσμα είναι αρνητικό . Αν είναι τότε το μετατρέπουμε σε θετικό με μια not και add +1 στο τέλος για να πάρουμε το συμπλήρωμα ως προς 2 , και επιστρέφουμε. Αν όχι συνεχίζουμε. Έχουμε έναν pointer που δείχνει στο πρώτο στοιχείο το C τον t2 αποθηκεύουμε το αποτέλεσμα που πήραμε σε θετικό αριθμό . Τέλος , αυξάνουμε τους pointer t0,t2 για να δείχνουν στον επόμενο (επειδή είναι words το αυξάνουμε κατά 4) και μειώνουμε τον pointer t1 για να δείχνει στον προηγούμενο του πίνακα B. Επαναλαμβάνουμε την

ίδια διαδικασία μέχρι να φτάσουμε στο τέλος του πίνακα (δηλαδή το κάνουμε Ν φορές).

2η Άσκηση: Weird_blinking

Κώδικας:

```
.globl main
.text
main:
//correct values:
/*
rotate_left:
00 01 02 04 08 10 20 40 80 81 82 84 88 90 A0 C0 C1
C2 C4 C8 D0 E0 E1 E2 E4 E8 F0 F1 F2 F4 F8 F9 FA FC FD FE FF
rotate_right:
FF 7F BF DF EF F7 FB FD FE 7E BE DE EE F6 FA FC 7C BC DC EC F4 F8 78 B8 D8 E8 F0 70 B0 D0
E0 60 A0 C0 40 80 00
*/
//we have to run the programm with break point 10/13/19/27/34/45
//lines and open memory in 0x80001400
    li a0,0 //a0 the led
    li a1,0x7 //init the number of times we move the led
    li a2,0x7
    lui a5,0x80001

    sw a0,1024(a5) //store the led value
rotate_left:
    add a0,a0,1 // add one to the led
    sw a0,1024(a5) //store it
    add a1,a2,0 //a1=a2 // init the number of slli we need to reach to the msb we want to
change
    add a3,zero,1 // init the add value we use to move the leds
how_many_times_left:
    add a0,a0,a3 // move the led
    slli a3,a3,1 // move the add value to correct the value of the leds
    sw a0,1024(a5) // store the led value
    add a1,a1,-1 //sub the number of slli we need
    bne a1,zero,how_many_times_left // do it for the number of slli in order to reach the c
                                    //orrect position


    add a2,a2,-1 // we have fixed the msb and now the move to the previous in line
    bne a2,zero,rotate_left // do it while all the leds are not fixed
```

```
add a0,a0,1 // add the last vaue to reach FF
sw a0,1024(a5) // store the ff
li a2,0x8 // add new value to fix the lights in different dir
rotate_right:
    add a1,a2,0 // init the times of we move the led until msb
    add a3,zero,0x80; //fix the sub value
how_many_times_right:
    sub a0,a0,a3 // we sub the correct number
    sw a0,1024(a5) //store it
    add a0,a0,a3 // add it again
    add a4,a3,0 // keep the last value of the sub value
    srli a3,a3,1 // move the sub value to correct position
    add a1,a1,-1 //sub the number of slli we need
    bne a1,zero,how_many_times_right// do it for the number of slli in order to reach the c
                                    //orrect position


    add a2,a2,-1// we have fixed the msb and now the move to the previous in line
    sub a0,a0,a4 //in the end just sub the last sub value to correct the number (last
iteration we dont want to add it again in the line 36)
    bne a2,zero,rotate_right// do it while all the leds are not fixed


sw a0,1024(a5) // show the 0x00
j main


.end
```

Περιγραφή της άσκησης:

- Θέλουμε να κάνουμε ένα rotation του 1 bit μεχρι να φτάσει στο τέλος . Η άσκηση είναι για 8 bits αριθμό για να μπορεί να εμφανιστεί σαν τιμή(Επεκτείνεται πολύ εύκολα άμα αλλάξουμε το a2, και το a3 στην πίσω επανάληψη) . Έχουμε το a2 το οποίο είναι το bit που θέλουμε να φτάσουμε και αυτό μειώνεται κάθε φορά που τοποθετούμε 1 στην θέση που δείχνει το a2 . Έχουμε ένα a1 το οποίο θα κάνει shift left τον 1 για να φτάσει στο msb . Αρχικά τα αρχικοποιούμε και τα 2 στο 0x07 κάνουμε το a3 =1 το προσθέτουμε στο a0 το οποίο το printαρουμε στην θέση 0x800014000 , κάνουμε slli το a3 κατά μια θέση , αφαιρούμε τον αριθμό στο a1 και μέχρι να φτάσει αυτός ο αριθμός στο 0 συνεχίζουμε. Όταν

φτάσει στο 0 έχουμε φτάσει στο msb που θέλαμε , μειώνουμε το a2 και μέχρι το a2 να φτάσει στο 0 συνεχίζουμε . Προσθέτουμε το τελευταίο 1 στο a0 για να φτάσει το 0xff . Ξεκρίναμε το rotate_right αρχικοποιώντας την τιμή . Τώρα αφαιρούμε το a3 γιατί θέλουμε να μεταφέρουμε την απώλεια 1 στο αποτέλεσμα. Αφαιρούμε λοιπόν το a3 από το a1 και το αποθηκεύουμε , το προσθέτουμε μετά πάλι και κάνουμε srli το a3 για να πάρουμε το επιθυμητό νούμερο . Το επαναλαμβάνουμε αυτό μέχρι να φτάσει το a1 στο 0 το οποίο το αφαιρούμε σε κάθε επανάληψη . Όταν τελειώσει η επανάληψη μειώνουμε κατά 1 το a2 και αφαιρούμε και από το a0 το τελευταίο a3 που προσθέσαμε γιατί δεν το θέλαμε, το χρησιμοποιήσαμε μόνο για την σωστή αναπαράσταση του αποτελέσματος. Επαναλαμβάνουμε την ίδια διαδικασία για a2 φορές μέχρι το αποτέλεσμα να γίνει 0x00.