ΠΑΝΕΠΙΣΤΗΜΙΟ ΘΕΣΣΑΛΙΑΣ – ΠΟΛΥΤΕΧΝΙΚΗ ΣΧΟΛΗ

ΤΜΗΜΑ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ Η/Υ

ΠΡΟΟΔΟΣ ΣΤΗΝ *ΟΡΓΑΝΩΣΗ ΣΤΟΥΣ Η/Υ (ΗΥ232)*

Σάββατο, 16 Νοεμβρίου 2013

ΔΙΑΡΚΕΙΑ ΔΙΑΓΩΝΙΣΜΑΤΟΣ 2,5 ΩΡΕΣ

1) (Προγραμματισμός MIPS assembly με αναδρομή, 44=12+22+10 μονάδες)

Η συνάρτηση *int strcmp(const char * s1, const char * s2)* συγκρίνει δύο null-terminated string s1 και s2 και επιστρέφει τις παρακάτω τιμές:

- 0 εάν s1==s2, δηλ. s1 και s2 έχουν το ίδιο μήκος και όλοι χαρακτήρες που τα αποτελούν είναι οι ίδιοι.
- θετικό αριθμό εάν s1>s2, δηλ. εάν για τους πρώτους χαρακτήρες c1 και c2 του s1 και s2 που διαφέρουν, ισχύει ASCII(c1) > ASCCI(c2)
- αρνητικό αριθμό εάν s1<s2, δηλ. εάν για τους πρώτους χαρακτήρες c1 και c2 του s1 και s2 που διαφέρουν, ισχύει ASCII(c1) < ASCCI(c2)

Για παράδειγμα, “ade” > “ada” , και “1AB” < “ab”. Η άσκηση αυτή σας ζητάει να υλοποιήσετε την συνάρτηση *strcmp* **με αναδρομή**. Λύσεις που δεν χρησιμοποιούν αναδρομή δεν θα γίνονται δεκτές. Πιο συγκεκριμένα, απαντήστε στις παρακάτω ερωτήσεις:

a) Να γραφεί η αναδρομική συνάρτηση *strcmp* σε κάποια γλώσσα υψηλού επιπέδου όπως η C. Αυτό θα σας βοηθήσει και στην επόμενη ερώτηση.
b) Να γραφεί η αναδρομική συνάρτηση *strcmp* σε MIPS assembly. Δεν είναι ανάγκη να γράψετε την συνάρτηση *main* που καλεί την *strcmp*, αλλά θα πρέπει να κρατήσετε όλες τις συμβάσεις που έχουν να κάνουν με την στοίβα της συνάρτησης.
c) Ποιοι είναι οι ελάχιστοι καταχωρητές που θα πρέπει σίγουρα να αποθηκεύονται στην στοίβα σε κάθε κλήση της συνάρτησης για να είναι ο κώδικας της assembly λειτουργικά σωστός; Ποιο είναι το μέγιστο μέγεθος που φθάνει η στοίβα στην περίπτωση που συγκρίνουμε δύο strings μήκους n και m, αντίστοιχα; Να αναφέρετε το μέγιστο μέγεθος ως συνάρτηση *f(n,m).*

Λύσεις

```
int strcmp (const char * s1, const char * s2) {
   if ((*s1 != *s2) || (*s1 == NULL))
          return (*s1-*s2);
   else
          return strcmp(s1+1,s2+1);
}
```

```
int strcmp (const char * s1, const char * s2) {
     if (*s1 > *s2)
          return 1;
     else  if (*s1 < *s2)
          return -1;
     else if (*s1 == NULL)
          return 0;
    else
          return strcmp(s1+1,s2+1);
}
```

```
strcmp_rec:
        addi $sp, $sp, -4
        sw $ra, 0($sp)

        lb $t0, 0($a0)
        lb $t1, 0($a1)
        bne $t0, $t1, L
        beqz $t0, L
        addi $a0, $a0, 1
        addi $a1, $a1, 1
        jal strcmp_rec
        lw $ra, 0($sp)
        addi $sp, $sp, 4
        jr $ra
L:
        sub $v0, $t0, $t1
        addi $sp, $sp, 4
        jr $ra
```

c) Χρειαζόμαστε τουλάχιστον τον $ra για να είναι το πρόγραμμα λειτουργικά σωστό. Η χειρότερη περίπτωση για το μέγεθος της στοίβας είναι min(n,m)*4 bytes.

2) (Προγραμματισμός MIPS assembly II, 32 μονάδες)

Να γραφεί κώδικας MIPS assembly ο οποίος θα παίρνει το λιγότερο σημαντικό byte του καταχωρητή $s0 και θα αναστρέφει τα bits του έτσι ώστε το λιγότερο σημαντικό bit θα μπει στην πιο σημαντική θέση του byte. Τα υπόλοιπα bytes του $s0 θα παραμείνουν ως έχουν. Το αποτέλεσμα θα γράφεται πάλι στον καταχωρητή $s0. Για παράδειγμα, εάν $s0 = 0xA45F90B4, ο κώδικας θα πρέπει να γράψει στον $s0 την τιμή 0xA45F902D.

Ο κώδικας σας δεν θα πρέπει να περιλαμβάνει εντολές αποθήκευσης ή φόρτωσης από την μνήμη. Μπορείτε να χρησιμοποιήσετε όσους βοηθητικούς καταχωρητές θέλετε στον κώδικά σας.

```
        move $t0, $s0
        andi $t0, $t0, 0xFF
        andi $s0, $s0, 0xFFFFFF00
        li $t2, 0
        li $t3, 7
loop:               # One bit at a time
        andi $t1, $t0, 0x1
        or $t2, $t2, $t1
        srl $t0,$t0, 1
        addi $t3, $t3, -1
        bltz $t3, Exit
        sll $t2, $t2, 1
        j loop
Exit:
        or $s0, $s0, $t2
```

3) Απόδοση Συστήματος (24=8+8+8 μονάδες)

Θεωρείστε δύο διαφορετικές μικρο-αρχιτεκτονικές υλοποιήσεις, P1 και P2, της αρχιτεκτονικής MIPS_SMALL. Έστω ότι υπάρχουν πέντε διαφορετικές κλάσεις εντολών στην MIPS_SMALL (A, B, C, D, E). Ο παρακάτω πίνακας που θα χρησιμοποιηθεί στα ερωτήματα αυτής της άσκησης δείχνει την συχνότητα του ρολογιού και το CPI (clocks per instruction) για κάθε κλάση εντολών της MIPS_SMALL.

Επεξεργαστής	Συχνότητα Ρολογιού	CPI A	CPI B	CPI C	CPI D	CPI E
P1	1.0 GHz	1	1	2	3	2
P2	1.5 GHz	1	2	3	4	3

a) Θεωρείστε ότι η μέγιστη απόδοση ενός επεξεργαστή βρίσκεται όταν ο επεξεργαστής εκτελέσει την πιο γρήγορη αλληλουχία εντολών. Ποια είναι η μέγιστη απόδοση (σε εντολές ανά sec) για τους επεξεργαστές P1 και P2;

b) Κατά την διάρκεια εκτέλεσης ενός προγράμματος ό ίδιος αριθμός εντολών εκτελείται από κάθε μία από τις 5 κλάσεις εντολών, εκτός από τις εντολές A που εκτελούνται δύο φορές πιο συχνά από τις άλλες. Ποιος από τους 2 επεξεργαστές είναι πιο γρήγορος και κατά πόσο;

c) Κατά την διάρκεια εκτέλεσης ενός προγράμματος ό ίδιος αριθμός εντολών εκτελείται από κάθε μία από τις 5 κλάσεις εντολών, εκτός από τις εντολές E που εκτελούνται δύο φορές πιο συχνά από τις άλλες. Ποιος από τους 2 επεξεργαστές είναι πιο γρήγορος και κατά πόσο;

Solution

a) Η πιο γρήγορη αλληλουχία εντολών είναι όταν οι επεξεργαστές εκτελούν εντολές κλάσης A που έχουν το μικρότερο CPI=1.
Χρόνος εκτέλεσης προγράμματος (P1) = IC * CPI * Clock Period = IC * 1 * (1/10^9)sec = IC * 10^{-9} sec = 10^{-9} sec/inst. Άρα απόδοση P1 = 10^9 inst/sec
Χρόνος εκτέλεσης προγράμματος (P2) = IC * 1 * (1/1.5*10^9) sec = IC * 0.67*10^{-9} . Αρα απόδοση P2 = 1.5*10^9 inst/sec

b) Θεωρούμε ότι εκτελούνται 2 εντολές A και από μία εντολή από όλες τις υπόλοιπες κλάσεις.
Χρόνος εκτέλεσης προγράμματος (P1) = IC * CPI * Clock Period = (2*1+1*1+1*2+1*3+1*2)* 10^{-9} = 10 * 10^{-9} = 10 ns
Χρόνος εκτέλεσης προγράμματος (P2) = IC * CPI * Clock Period = (2*1+1*2+1*3+1*4+1*3)* 0.67*10^{-9} = 9.33 ns
ET(P1)/ET(P2) = 1,072. Άρα ο επεξεργαστής P2 είναι 1.072 φορές πιο γρήγορος.

c) Θεωρούμε ότι εκτελούνται 2 εντολές E και από μία εντολή από όλες τις υπόλοιπες κλάσεις.
Χρόνος εκτέλεσης προγράμματος (P1) = IC * CPI * Clock Period = (1*1+1*1+1*2+1*3+2*2)* 10^{-9} = 9 * 10^{-9} = 11 ns
Χρόνος εκτέλεσης προγράμματος (P2) = IC * CPI * Clock Period = (1*1+1*2+1*3+1*4+2*3)* 0.67*10^{-9} = 10.67 ns
ET(P1)/ET(P2) = 1,03. Άρα ο επεξεργαστής P2 είναι 1.03 φορές πιο γρήγορος.