1Η ΕΡΓΑΣΙΑ ΕΡΓΑΣΤΗΡΙΟΥ ΒΑΣΙΚΑ ΘΕΜΑΤΑ ΑΡΧΙΤΕΚΤΟΝΙ
ΥΠΟΛΟΓΙΣΤΩΝ

10

Συγγραφείς: Βέργος Γεώργιος ΑΜ :1072604 , Τσούλος Βασίλειος ΑΜ:1072605.

ΠΙΝΑΚΑΣ ΤΙΜΩΝ ΤΩΝ ΚΑΤΑΧΩΡΗΤΩΝ R0-R6 ΚΑΙ PC

	R0	R1	R2	R3	R4	R5	R6	PC
E0	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00008004
E1	0x00000020	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00008008
E2	0x00000020	0x00000080	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x0000800C
E3	0x00000020	0x00000080	0xFFFFEFFF	0x00000000	0x00000000	0x00000000	0x00000000	0x00008010
E4	0x00000020	0x00000080	0xFFFFEFFF	0x00009000	0x00000000	0x00000000	0x00000000	0x00008014
E5	0x00000020	0x00000080	0xFFFFEFFF	0x00009004	0xCAFEBABA	0x00000000	0x00000000	0x00008018
E6	0x00000020	0x00000080	0xFFFFEFFF	0x00009006	0xCAFEBABA	0x00000078	0x00000000	0x0000801C
E7	0x00000020	0x00000080	0xFFFFEFFF	0x00009008	0xCAFEBABA	0x00000078	0xFFFF8234	0x00008020
E8	0x00000020	0x00000080	0xFFFFEFFF	0x00009008	0xCAFEBABA	0x00000078	0xFFFF8234	0x00008024
E9	0x00000020	0x00000080	0xFFFFEFFF	0x00009020	0xCAFEBABA	0x00000078	0xFFFF8234	0x00008028
E10	0xCAFEBABA	0x00000078	0xFFFF8234	0x00009014	0xCAFEBABA	0x00000078	0xFFFF8234	0x0000802C
E11	0xCAFEBABA	0x00000078	0xFFFF8234	0x00009008	0x00000020	0x00000080	0xFFFFEFFF	0x00008030

Σημείωση:

Ως αρχική τιμή στα data βάζω την τιμή 0x8000 . Έτσι κατά το debugging του προγράμματος χρειάζεται πατώντας F2 να αλλάξω την τιμή του

καταχωρητή(μετρητή προγράμματος) ,PC(R15) σε 0x00008000 από την 0x00000000.

Ο ΚΩΔΙΚΑΣ ΤΗΣ ΑΣΚΗΣΗΣ ΚΑΙ ΟΙ ΕΞΗΓΗΣΕΙΣ ΤΩΝ ΓΡΑΜΜΩΝ ΤΟΥ

0) .arm @Σηματοδοτεί την έναρξη του κώδικα και πρέπει πάντα να τοποθετείται στην αρχή του προγράμματος. Ενημερώνουμε τον assembler να παράγει κώδικα των 32 δυαδικών ψηφίων.

1) .text @Τοποθετεί τον ακόλουθο κώδικα σε συγκεκριμένη θέση

2) .global main @ Η ντιρεκτίβα global ορίζει την ετικέτα main ως καθολική.

main:

3) STMDB R13!, {R0-R12, R14} Αποθήκευση των καταχωρητών που θα χρησιμοποιήσουμε στη σωρό του συστήματος.

4) MOV R0, #0x20 @Μεταφέρει στον καταχωρητή R0 την δεκαξαδική τιμή 00000020.

5) MOV R1, R0, LSL #2 @Μεταφέρει το περιεχόμενο του R0 λογικά ολισθημένο αριστερά κατά δύο θέσεις και αποθηκεύει το αποτέλεσμα στον R1

6) MVN R2, R1, LSL #1 @Παίρνει το περιεχόμενο του R1 του κάνει λογική ολίσθηση αριστερά κατά μία θέση αντιστρέφει την τιμή αυτή

κάνοντας τα 1 σε 0 και τα 0 σε 1 και αποθηκεύει το αποτέλεσμα στον R2.

7) LDR R3, =Values @Μεταφέρουμε στον καταχωρητή R3 τη διεύθυνση που σηματοδοτεί η ετικέτα Values(Θα είναι η τιμή 0x9000 γιατί έχουμε ρυθμίσει ως data start την 0x9000 στις ρυθμίσεις).

8) LDR R4, [R3], #4 @Μεταφέρει το περιεχόμενο της θέσης μνήμης με διέυθυνση το περιεχόμενο του καταχωρητή R3 στον καταχωρητή R4 και αποθηκεύει την τιμή <τρέχον περιεχόμενο του R3 + 4> στον R3.

9) LDRB R5, [R3], #2 @Μεταφέρει 1 byte του περιεχόμενου της θέσης μνήμης με διεύθυνση το περιεχόμενο του καταχωρητή R3 στον καταχωρητή R5 και αποθηκεύει την τιμή <τρέχον περιεχόμενο του R3 + 2 >στον R3.

10) LDRSH R6, [R3], #2 @Μεταφέρει 1 halfword(16bits -τα 16 λιγότερο σημαντικά) του περιεχόμενου της θέσης μνήμης με διεύθυνση το περιεχόμενο του καταχωρητή R3 (στο οποίο έχει πραγματοποιηθεί επέκταση προσήμου λόγω του S)στον καταχωρητή R6 και αποθηκεύει την τιμή <τρέχον περιεχόμενο του R3 + 2> στον R3.

11) LDR R3, =Stack @Μεταφέρουμε στον καταχωρητή R3 τη διεύθυνση που σηματοδοτεί η ετικέτα Values.

12) STMIA R3!, {R0-R2, R4-R6} @ Το πρόγραμμα θα βάλει στη θέση μνήμης με διεύθυνση το τρέχον περιεχόμενο του καταχωρητή R3 το περιεχόμενο του καταχωρητή R0 , έπειτα θα βάλει στη θέση μνήμης με διεύθυνση το τρέχον περιεχόμενο του καταχωρητή R3 +4 το περιεχόμενο του καταχωρητή R1, έπειτα θα βάλει στη θέση μνήμης με

διεύθυνση το τρέχον περιεχόμενο του καταχωρητή R3 +8 το περιεχόμενο του καταχωρητή R2, έπειτα θα βάλει στη θέση μνήμης με διεύθυνση το τρέχον περιεχόμενο του καταχωρητή R3 +C το περιεχόμενο του καταχωρητή R4,

έπειτα θα βάλει στη θέση μνήμης με διεύθυνση το τρέχον περιεχόμενο του καταχωρητή R3 +10(HEX) το περιεχόμενο του καταχωρητή R5 και τέλος θα βάλει στη θέση μνήμης με διεύθυνση το τρέχον περιεχόμενο του καταχωρητή R3 +14(HEX) το περιεχόμενο του καταχωρητή R6.

Η ύπαρξη του θαυμαστικού δίπλα από τον καταχωρητή R3 δηλώνει πως θα ανανεωθεί το περιεχόμενο του R3 μετά την εκτέλεση της εντολής από την προηγούμενη τιμή του(προ-εκτέλεσης της εντολής +18(δεκαεξαδικό)=4*6(Ο R3 θεωρείται καταχωρητής βάσης). Το IA(increment after) στην εντολή STMIA δηλώνει πως η διεύθυνση προσπέλασης θα αυξηθεί κατά 4 ΑΦΟΥ γίνει πρώτα η προσπέλαση στη συγκεκριμένη θέση μνήμης.

13) LDMDB R3!, {R0-R2} @ Πάλι με την προσθήκη του θαυμαστικού(!) ενεργοποιείται η τροποποιήση του καταχωρητή βάσης(του R3). Επειδή αυτή τη φορά οι προσπελάσεις που θα πραγματοποιήσει το πρόγραμμα είναι decrement περιμένουμε μετά την εκτέλεση αυτής της εντολής η τιμή του καταχωρητή R3 να είναι <τρέχουσα τιμή του R3> -4*3(δεκαεξαδικό).Εφόσον η εντολή είναι τύπου decrement before(DB) πριν προσπελαστεί η αντίστοιχη θέση στη μνήμη η διεύθυνση προσπέλασης μειώνεται κατά 4. Η εντολή είναι τύπου LDM οπότε έχω μεταφορά δεδομένων από τη μνήμη αυτή τη φορά σε καταχωρητές. Η διαδικασία είναι η ακόλουθη: Το πρόγραμμα θα βάλει τα δεδομένα

της θέσης μνήμης με διεύθυνση <τρέχον περιεχόμενο του R3> - 4 στον καταχωρητή R2, έπειτα θα βάλει τα δεδομένα της θέσης μνήμης με διεύθυνση <τρέχον περιεχόμενο του R3> - 8 στον καταχωρητή R1

Και τέλος θα βάλει τα δεδομένα της θέσης μνήμης με διεύθυνση <τρέχον περιεχόμενο του R3> - C στον καταχωρητή R0.

14) LDMDB R3!, {R4-R6} @ Πάλι με την προσθήκη του θαυμαστικού(!) ενεργοποιείται η τροποποιήση του καταχωρητή βάσης(του R3). Επειδή αυτή τη φορά οι προσπελάσεις που θα πραγματοποιήσει το πρόγραμμα είναι decrement περιμένουμε μετά την εκτέλεση αυτής της εντολής η τιμή του καταχωρητή R3 να είναι <τρέχουσα τιμή του R3> - 4*3=C(δεκαεξαδικό). Εφόσον η εντολή είναι τύπου decrement before(DB) πριν προσπελαστεί η αντίστοιχη θέση στη μνήμη η διεύθυνση προσπέλασης μειώνεται κατά 4. Η εντολή είναι τύπου LDM οπότε έχω μεταφορά δεδομένων από τη μνήμη αυτή τη φορά σε καταχωρητές. Η διαδικασία είναι η ακόλουθη: Το πρόγραμμα θα βάλει τα δεδομένα της θέσης μνήμης με διεύθυνση <τρέχον περιεχόμενο του R3> - 4 στον καταχωρητή R6, έπειτα θα βάλει τα δεδομένα της θέσης μνήμης με διεύθυνση <τρέχον περιεχόμενο του R3> - 8 στον καταχωρητή R5 και τέλος θα βάλει τα δεδομένα της θέσης μνήμης με διεύθυνση <τρέχον περιεχόμενο του R3> - C στον καταχωρητή R4.

15) LDMIA R13!, {R0-R12, PC} @Πάλι αυτή η εντολή μεταφέρει δεδομένα από τη μνήμη σε καταχωρητές. Με την ύπαρξη του θαυμαστικού επιτρέπουμε την τροποποιήση του περιεχομένου του καταχωρητή R13. Πρόκειται για εντολή increment after δηλαδή η διεύθυνση προσπέλασης στη μνήμη αυξάνεται κατά 4 αφού γίνει προσπέλαση στη μνήμη. Έτσι αναμένουμε το περιεχόμενο του καταχωρητή R13 μετά την εκτέλεση της εντολής να είναι <τρέχον περιεχόμενο του R13> +4*14(=38 hex). Έτσι το πρόγραμμα θα βάλει το

περιεχόμενο της θέσης μνήμης με διεύθυνση το τρέχον περιεχόμενο του R13 στον καταχωρητή R0 , έπειτα θα βάλει το περιεχόμενο της θέσης μνήμης με διεύθυνση το τρέχον περιεχόμενο του R13 +4 στον καταχωρητή R1 , έπειτα θα βάλει το περιεχόμενο της θέσης μνήμης με διεύθυνση το τρέχον περιεχόμενο του R13 +8 στον καταχωρητή R2, έπειτα θα βάλει το περιεχόμενο της θέσης μνήμης με διεύθυνση το τρέχον περιεχόμενο του R13 +C στον καταχωρητή R3, έπειτα θα βάλει το περιεχόμενο της θέσης μνήμης με διεύθυνση το τρέχον περιεχόμενο του R13 +10 στον καταχωρητή R4, έπειτα θα βάλει το περιεχόμενο της θέσης μνήμης με διεύθυνση το τρέχον περιεχόμενο του R13 +14 στον καταχωρητή R5, έπειτα θα βάλει το περιεχόμενο της θέσης μνήμης με διεύθυνση το τρέχον περιεχόμενο του R13 +18 στον καταχωρητή R6, έπειτα θα βάλει το περιεχόμενο της θέσης μνήμης με διεύθυνση το τρέχον περιεχόμενο του R13 +1C στον καταχωρητή R7, έπειτα θα βάλει το περιεχόμενο της θέσης μνήμης με διεύθυνση το τρέχον περιεχόμενο του R13 +20 στον καταχωρητή R8 , έπειτα θα βάλει το περιεχόμενο της θέσης μνήμης με διεύθυνση το τρέχον περιεχόμενο του R13 +24 στον καταχωρητή R9, έπειτα θα βάλει το περιεχόμενο της θέσης μνήμης με διεύθυνση το τρέχον περιεχόμενο του R13 +28 στον καταχωρητή R10, έπειτα θα βάλει το περιεχόμενο της θέσης μνήμης με διεύθυνση το τρέχον περιεχόμενο του R13 +2C στον καταχωρητή R11 , έπειτα θα βάλει το περιεχόμενο της θέσης μνήμης με διεύθυνση το τρέχον περιεχόμενο του R13 +30 στον καταχωρητή R12 και τέλος θα βάλει το περιεχόμενο της θέσης μνήμης με διεύθυνση το τρέχον περιεχόμενο του R13 +34 στον καταχωρητή PC. Επειδή κατά την εκτέλεση του προγράμματος οι περιοχές της μνήμης με τις παραπάνω διευθύνσεις έχουν ως δεδομένα την τιμή μηδέν(0) μετά την εκτέλεση της εντολής αυτής όλοι οι καταχωρητές μηδενίζονται.

.data @Ενημερώνουμε τον συμβολομεταφραστή πως τα δεδομένα που θα ακολουθήσουν(τα 0xCAFEBABA και 0x82345678) πρέπει να τα τοποθετήσει στο τμήμα της μνήμης που δεσμεύει για να τοποθετεί μη-σταθερά δεδομένα(Αυτό το τμήμα μνήμης σηματοδοτείται από την ετικέτα Values). Τα δεδομένα αυτά αποθηκεύονται σε διαδοχικές θέσεις μνήμης κατά little endian. Τα αποθηκεύει δηλαδή στην περιοχή data που ορίζει το λειτουργικό σύστημα για τα δεδομένα των προγραμμάτων.
Values: @ Τα παρακάτω words τοποθετούνται στη περιοχή της μνήμης που σηματοδοτείται από την ετικέτα Values.

.word 0xCAFEBABA @Ντιρεκτίβα word. Ενημερώνω τον συμβολομεταφραστή πως ακολουθεί μια σειρά απο words(τιμές των 32 bits τις οποίες θα αποθηκεύει στη μνήμη σε διαδοχικές θέσεις.

.word 0x82345678

Stack: @ Τα παρακάτω words τοποθετούνται στη περιοχή της μνήμης που σηματοδοτείται από την ετικέτα Stack. Ορίζω μια σειρά από τιμές(όλες τους 0) οι οποίες διαχωρίζονται μεταξύ τους με κόμμα και θα τοποθετηθούν στην μνήμη(όπου σηματοδοτεί η Stack) σε διαδοχικές θέσεις, ξεκινώντας από την πρώτη τιμή, η οποία θα τοποθετηθεί στην μικρότερη διεύθυνση.

.word 0,0,0,0

.word 0,0,0,0