ΕΡΓΑΣΤΗΡΙΟ ΨΗΦΙΑΚΗΣ ΕΠΕΞΕΡΓΑΣΙΑΣ ΣΗΜΑΤΩΝ

Εργαστήριο 1

Ομάδα 09 – Group 1

ΕΠΩΝΥΜΟ	ΟΝΟΜΑ	ΑΡΙΘΜΟΣ ΜΗΤΡΩΟΥ
ΔΑΣΟΥΛΑΣ	ΙΩΑΝΝΗΣ	1053711
ΔΟΥΡΔΟΥΝΑΣ	ΑΡΙΣΤΕΙΔΗΣ ΑΝΑΡΓΥΡΟΣ	1047398

Προγραμματισμός του C67x σε assembly(1/2)

Στόχος:

Στόχος της άσκησης είναι η εισαγωγή στην αρχιτεκτονική και το σύνολο εντολών του C67X με σκοπό την εξοικείωση με τις εσωτερικές λειτουργίες της ΚΜΕ και του προγραμματισμού με χρήση γλώσσας assembly. Επίσης, γίνεται εξάσκηση στις βασικές τεχνικές προγραμματισμού χρησιμοποιώντας τον assembler της TI.

Ασκήσεις:

Άσκηση 1.1

Πρόσθεση και πολλαπλασιασμός. Να γραφτεί ένα πρόγραμμα σε γλώσσα assembly που να υπολογίζει το A3*(A4+A5)+A2*A1.

Πρόγραμμα:

1. MVK 0x1234, A1
2. MVK 0x0012, A2
3. MVK 0x0020, A3
4. MVK 0x0008, A4
5. MVK 0x0400, A5

6. ADD A4,A5,A4
7. MPY A2,A1,A1
8. MPY A3,A4,A3
9. NOP
10. ADD A3,A1,A1

Στις πρώτες 5 γραμμές φορτώνονται στους καταχωρητές A1 έως A5 τιμές. Έπειτα, σύμφωνα με το εγχειρίδιο TMS320C67x/C67x+ DSP CPU and Instruction Set η εντολή ADD που εκτελεί την πρόσθεση απαιτεί 1 κύκλο ρολογιού, ενώ η MPY που εκτελεί

τον πολλαπλασιασμό απαιτεί 2 κύκλους ρολογιού, άρα 1 delay slot αν ακολουθεί άμεση χρησιμοποίηση του αποτελέσματος από το πρόγραμμα. Για αυτόν τον λόγο εισάγεται μία nop οδηγία στην γραμμή 9 για να καταχωρηθεί σωστά το αποτέλεσμα του πολλαπλασιασμού της γραμμής 8 στον καταχωρητή A3 και έπειτα να γίνει η τελική πρόσθεση στην γραμμή 10.

Άσκηση 1.2

Να γραφτεί το αρχείο διασύνδεσης εντολών του eclass ως dsk6713.cmd και σωθεί στον κατάλογο αρχείων. Για να φορτωθεί ο κώδικας προγράμματός στη μνήμη flash ποιες αλλαγές πρέπει να γίνουν στο αρχείο;

Προκειμένου να τρέξει ο κώδικας Assembly στην KME πρέπει πρώτα να υποδειχθεί η θέση μνήμης στην οποία στην οποία θα φορτωθεί ο κώδικα. Η C67X KME έχει χώρο στην εσωτερική μνήμη για να αποθηκεύσει τον κώδικα και τα δεδομένα ενός προγράμματος. Το DSK board έχει εξωτερική RAM η οποία μπορεί να επιλεχθεί για να φορτωθεί ο κώδικας. Ο χάρτης μνήμης δίνεται στη συνέχεια. Αφού υποδειχθεί η διεύθυνση της μνήμης στην οποία θα φορτωθεί ο κώδικας για να εκτελεστεί , δηλαδή να κάνει assemble, δημιουργείται ένας μεταθέσιμος κώδικας. Αυτό σημαίνει ότι ο κώδικας δεν έχει κάποια σταθερή αναφορά μνήμης και μπορεί να τοποθετηθεί σε οποιαδήποτε θέση μνήμης εφ' όσον έχουν υποδειχθεί σχετικές πληροφορίες για το πού θα τοποθετηθεί στο χάρτη μνήμης.

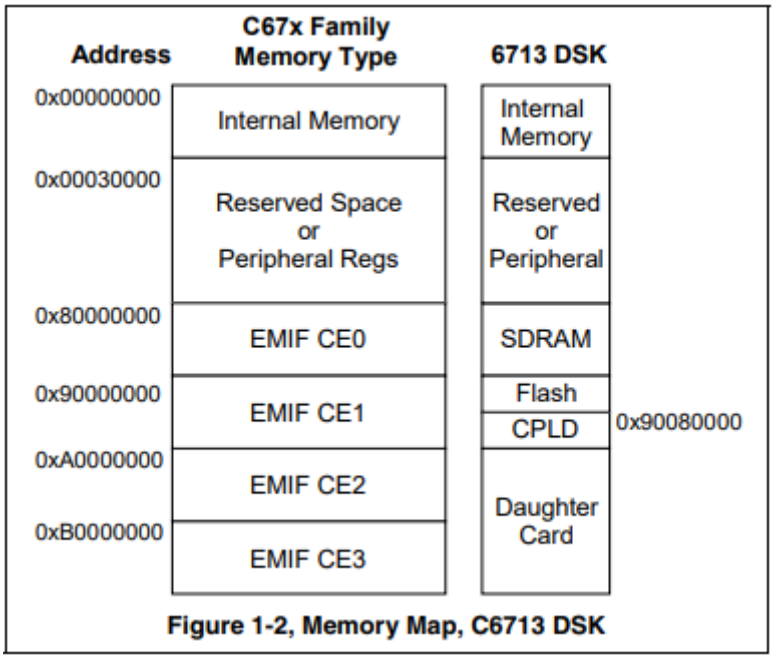

Figure 1-2, Memory Map, C6713 DSK

Στη συνέχεια ο linker συνδυάζει τα διαφορετικά κομμάτια του Assembled κώδικα για να παράσχει τον εκτελέσιμο κώδικα. Ο εκτελέσιμος κώδικας έχει όλες τις πληροφορίες για τις θέσης μνήμης. Για να είναι σε θέση ο linker να δώσει έναν εκτελέσιμο κώδικα με πραγματικό προσδιορισμό των θέσεων μνήμης, κώδικα και δεδομένων, πρέπει να δώσουμε στον linker το χάρτη μνήμης (φυσικές διευθύνσεις) του DSK board

Το αρχείο dsk6713.cmd είναι το ακόλουθο :

```
1.  MEMORY
2.  {
3.  VECS: org = 0h, len = 0x220
4.  I_HS_MEM: org = 0x00000220, len = 0x00000020
5.  IRAM: org = 0x00000240, len = 0x0000FDC0
6.  SDRAM: org = 0x80000000, len = 0x01000000
7.  FLASH: org = 0x90000000, len = 0x00020000
8.  }
9.
10. SECTIONS
11. {
12. /* Created in vectors.asm */
13. vectors :> VECS
14. /* Created by Assembler */
15. .text :> IRAM
16. }
```

Το αρχείο του χάρτη αποτελείται από δύο τμήματα. Το πρώτο μέρος τμήμα MEMORY και αναφέρεται στον χάρτη διευθύνσεων της ΚΜΕ. Το τμήμα VECS ξεκινάει από την φυσική διεύθυνση 0h και πάει μέχρι 0x220h (οι διευθύνσεις δίνονται σε δεκαεξαδική αναπαράσταση) και περιέχει το reset και τα διανύσματα διακοπής. Οι υπόλοιπες διευθύνσεις της εσωτερικής μνήμης βρίσκονται στο τμήμα με όνομα IRAM. Τέλος η τα τμήματα διευθύνσεων της εξωτερικής RAM και της εξωτερικής μνήμης FLASHROM έχουν τα ονόματα SDRAM και FLASH.

Το δεύτερο τμήμα ονομάζεται SECTIONS. Το τμήμα αυτό καθορίζει σε ποια διεύθυνση θα φορτωθεί το κάθε κομμάτι του κώδικα. Η γραμμή 13 δείχνει ότι το τμήμα διανυσμάτων (που καθορίζεται στο αρχείο vectors.asm) φορτώνεται αρχικά στη διεύθυνση μνήμης VECS (που αρχίζει στη 0x00000000). Όλα τα άλλα τμήματα παράγονται είτε από τον assembler είτε από το μεταγλωττιστή C.

Για να φορτωθεί ο κώδικας στην FLASHROM πρέπει να γίνει αντικατάσταση στη γραμμή 13 του IRAM με το όνομα της επιθυμητής θέσης μνήμης. Στην συγκεκριμένη περίπτωση FLASHROM μετατρέποντας την εντολή σε: .text :> FLASH

Άσκηση 1.3

Να γραφτεί και να αποθηκευτεί αρχείο ‘vectors.asm’.

Πρόγραμμα:

```
1.  .title  "vectors.asm"


2.  .ref      entry
```

```
3.  .sect   "vectors"


4.  rst:    mvkl    .s2     entry, b0
5.          mvkh    .s2     entry, b0
6.          b       .s2     b0
7.          nop
8.          nop
9.          nop
10.         nop
11.         nop
```

Η πρώτη γραμμή ονομάζει αυτό το κομμάτι του κώδικα ως vectors.asm. Η .ref directive παραθέτει τα συμβολικά ονόματα που ορίζονται σε ένα άλλο αρχείο και χρησιμοποιούνται στο παρόν αρχείο. Δηλαδή δηλώνει ότι το entry είναι ένα συμβολικό όνομα που ορίζεται αλλού (η .ref είναι παρόμοια με την extern δήλωση στη C). Η directive .sect απλά λέει ότι ο linker πρέπει να φορτώσει τις ακόλουθες assembly εντολές στο τμήμα vectors που ορίζεται στο αρχείο διασύνδεσης εντολών. Επειδή το παραπάνω αρχείο διασύνδεσης εντολών ορίζει το τμήμα vectors να αρχίζει στη διεύθυνση μνήμης 0x00000000, οι εντολές assembly φορτώνονται αρχικά σε αυτήν την θέση.

Όταν η C67X DSP CPU λαμβάνει το σήμα reset, η KME πρώτα αρχικοποιεί όλους τους καταχωρητές και στη συνέχεια αρχίζει την εκτέλεση του κώδικα στη διεύθυνση 0x00000000. Έτσι αρχικά πρέπει να φορτωθεί ο reset κώδικα στη διεύθυνση 0x00000000 πριν τρέξει οποιοδήποτε άλλος κώδικας. Το αρχείο vectors.asm είναι το κομμάτι του κώδικα που ορίζεται στον linker να φορτώσει σε αυτήν την διεύθυνση. Όταν έχει ονομαστεί το σημείο εισόδου του προγράμματός entry, τότε στο reset κατευθύνουμε την εκτέλεση σε αυτό το σημείο εισόδου. Οι γραμμές 4 και 5 στο παραπάνω vectors.asm φορτώνουν στον B0 καταχωρητή τη διεύθυνση μνήμης της εισόδου (entry) για να κάνει άλμα (branch) στη διεύθυνση που περιέχει ο B0 χρησιμοποιώντας την εντολή B (branch) στη γραμμή 6. Λόγω της παραλληλίας στην λειτουργία του επεξεργαστή (pipeline), πριν κάθε διακλάδωση, χρειάζονται πέντε οδηγίες nop (γραμμές 7 έως 11) μετά από κάθε εντολή b, εκτός αν θέλουμε να εκτελέσουμε πρόσθετες εντολές. Επίσης, ο κώδικας που εκτελεί το reset θα πρέπει να αποτελείται από συνολικά 8 εντολές assembly.

Άσκηση 1.4

Ο κώδικας που αναπτύχθηκε κατά την διάρκεια του εργαστηρίου είναι ο ακόλουθος:

```
1.          .def entry
```

```
2.              .text
3. entry:       MVK .S1 0x1234,A0
4.              MVK .S1 0x0012, A1
5.              ADD .L1 A0, A1, A2
6.              IDLE
7.              .end
```

Οι εντολές MVK χρησιμοποιούνται για να φορτώσουμε στους καταχωρητές A0 , A1 τις επιθυμητές τιμές (0x1234 , 0x0012) και η εντολή ADD προσθέτει τις τιμές των δύο καταχωρητών σε έναν νέο καταχωρητή A2. Τα .S1 , .L1 που προστέθηκαν σε σχέση με τον κώδικα της εκφώνησης της εργαστηριακής άσκησης αναφέρονται στις λειτουργικές μονάδες που επιτελούν τις συγκεκριμένες εντολές. Πιο συγκεκριμένα, η .L1 επιτελεί λογικές πράξεις μέχρι 32 bit και γ .S1 μεταφέρει πληροφορία από και προς τους καταχωρητές. Τέλος , η εντολή IDLE κρατάει την ΚΜΕ σε κατάσταση μη απασχόλησης μετά την εκτέλεση της εντολής ADD.

Άσκηση 1.5

Τροποποίηση του προγράμματος στο e-class για να υπολογίζει το ax+b. Έστω x = 2 και ότι είναι αποθηκευμένο στον A3 καταχωρητή.

Το πρόγραμμα που δίνεται στο e-class είναι το εξής:

```
1.  a       .set 0x4530
2.  b       .set 0x0031
3.  x       .set 0x2


4.          .def entry
5.          .text

6.  entry:  MVK a,A0
7.          MVK b,A1
8.          MVK x,A3
9.          MPY A0,A3,A4
10.         ADD A4,A1,A2
11.         IDLE
12.         .end
```

Αν εκτελεστεί το πρόγραμμα αυτό, δε θα δοθεί το επιθυμητό αποτέλεσμα. Ο λόγος είναι ότι στην γραμμή 9 εκτελείται πολλαπλασιασμός το αποτέλεσμα του οποίου χρησιμοποιείται στην γραμμή 10. Όπως εξηγήθηκε, όμως, σε παραπάνω ερώτημα, για να προκύψει σωστό αποτέλεσμα είναι απαραίτητο ένα delay slot ανάμεσα στις γραμμές 9 και 10. Το σωστό πρόγραμμα θα είναι έτσι:

```
1.  a       .set 0x4530
2.  b       .set 0x0031
```

```
3.  x       .set 0x2


4.      .def entry
5.      .text

6.  entry:  MVK a,A0
7.          MVK b,A1
8.          MVK x,A3
9.          MPY A0,A3,A4
10.         nop
11.         ADD A4,A1,A2
12.         IDLE
13.         .end
```

Για να βελτιωθεί ο κώδικας και να αποφευχθεί η χρήση της nop οδηγίας, αμέσως μετά από την φόρτωση των καταχωρητών με τις επιθυμητές τιμές, μπαίνει η εντολή πολλαπλασιασμού MPY. Στη συνέχεια φορτώνεται η τιμή b σε έναν καταχωρητή , καθώς δεν είναι απαραίτητη για κάποιον προηγούμενο υπολογισμό. Ως αποτέλεσμα, η εντολή MPY ,η οποία χρειάζεται δυο κύκλους ρολογιού για να εκτελεστεί , προλαβαίνει να αποθηκεύσει στον καταχωρητή A4 το επιθυμητό αποτέλεσμα. Το πρόγραμμα στη τελική του μορφή θα είναι το ακόλουθο:

```
1.  a       .set 0x4530
2.  b       .set 0x0031
3.  x       .set 0x2


4.      .def entry
5.      .text

6.  entry:  MVK a,A0
7.          MVK x,A3
8.          MPY A0,A3,A4
9.          MVK b,A1
10.         ADD A4,A1,A2
11.         IDLE
12.         .end
```

Βιβλιογραφία:

- **TMS320C67x/C67x+ DSP CPU and Instruction Set Reference Guide**
- **ΕΚΦΩΝΗΣΗ ΑΣΚΗΣΗΣ LAB-1 – ECLASS**
- **TMS320C6713 DATASHEET**