ΕΡΓΑΣΤΗΡΙΟ ΨΗΦΙΑΚΗΣ ΕΠΕΞΕΡΓΑΣΙΑΣ ΣΗΜΑΤΩΝ

## Εργαστήριο 2

## Ξεκινώντας τον προγραμματισμό σε συμβολική γλώσσα

|  |
| --- |
| Ομάδα 17 19-10-2018 & 26-10-2018 |
| Ασημακόπουλος Κωνσταντίνος 1046966 |
| Λουκαρέας Παύλος 1046970 |

**Περίληψη**

Σε αυτήν την άσκηση συνεχίσαμε με τον προγραμματισμό σε συμβολική γλώσσα στον επεξεργαστή C67X της TI. Αρχικά έγινε μια εξοικείωση με τον τρόπο εγγραφής και ανάγνωσης δεδομένων από την κύρια μνήμη και την φόρτωση σταθερών στους καταχωρητές. Έπειτα, μάθαμε για την οργάνωση των δεδομένων στη μνήμη και τρόπους διευθυνσιοδότησης, τα delay slots και την εντολή διακλάδωσης και εκτέλεση πράξεων υπό συνθήκη. Τέλος μάθαμε πως να ελέγχουμε τα περιφερειακά LEDs και switches του board.

Άσκηση 2.1

Σε αυτό το ερώτημα είδαμε πως να φορτώσουμε σταθερές σε καταχωρητές αναλόγως το μέγεθος τους και την τιμή τους ώστε να γίνει η σωστή επέκταση πρόσημου μέσω των εντολών MVK, MVKL και MVKH. Ο κώδικας για την φόρτωση είναι:

MVKL .S1 0xff15, A1

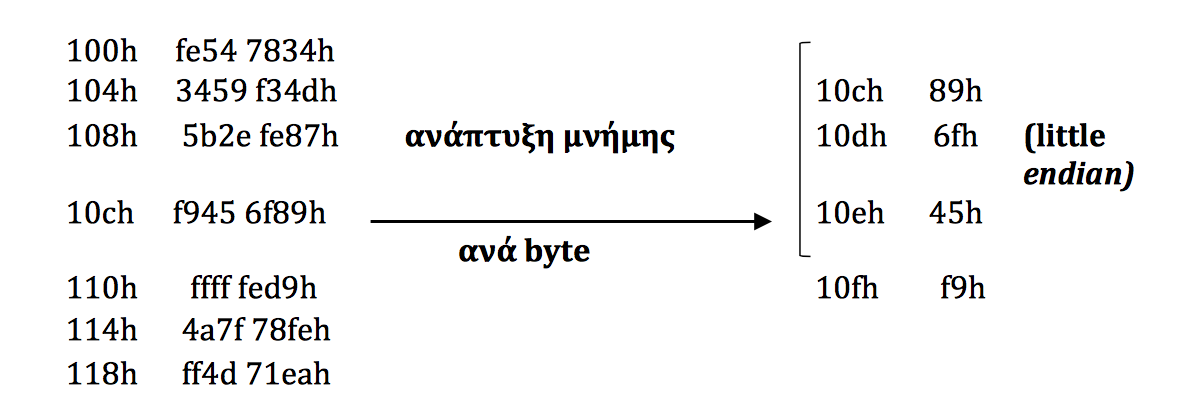
MVKH .S1 0xff15, A1

Πρέπει να χρησιμοποιήσουμε τις δύο αυτές εντολές ώστε η σταθερά να μην λάβει αρνητικό πρόσημο λόγω της επέκτασης πρόσημου που κάνει η εντολή MVK επειδή η σταθερά ξεκινάει με ‘1’ στο δυαδικό. Για την 32-bit σταθερά:

MVKL .S2 0xabcd45df, B0

MVKH .S2 0xabcd45df, B0

Άσκηση 2.2

Σε αυτό το ερώτημα υποθέτουμε ότι ο καταχωρητής Α10 = 0000 0108h. Οι τιμές στη μνήμη μας δίνονται στην εκφώνηση 

Μας ζητείται η τιμή του των καταχωρητών του Α1 και Α10 σε κάθε μία από τις παρακάτω περιπτώσεις

1. LDW .D1 \*A10, A1

Εδώ φορτώνεται στον Α1 η τιμή 5b2efe87h η οποία βρίσκεται στη διεύθυνση που δείχνει ο Α10 του οποίου η τιμή παραμένει σταθερή.

1. LDΗ .D1 \*A10, A1

Εδώ φορτώνεται στον Α1 η τιμή fe87h η οποία βρίσκεται στη διεύθυνση ξεκινάει από τη θέση που δείχνει ο Α10 και φορτώνονται τα 16 LSB λόγω της little endian. Η τιμή του Α10 παραμένει σταθερή.

1. LDΒ .D1 \*A10, A1

Εδώ φορτώνεται στον Α1 η τιμή 87h η οποία βρίσκεται στη διεύθυνση ξεκινάει από τη θέση που δείχνει ο Α10 και φορτώνονται τα 8 LSB λόγω της little endian. Η τιμή του Α10 παραμένει σταθερή.

1. LDW .D1 \*-A10[1], A1

Εδώ φορτώνεται στον Α1 η τιμή 3459f34dh η οποία βρίσκεται στη διεύθυνση που δείχνει ο Α10 μείον μια λέξη. Η τιμή του Α10 παραμένει σταθερή.

1. LDW .D1 \*+A10[1], A1

Εδώ φορτώνεται στον Α1 η τιμή f9456f89h η οποία βρίσκεται στη διεύθυνση που δείχνει ο Α10 συν μια λέξη. Η τιμή του Α10 παραμένει σταθερή.

1. LDW .D1 \*+A10[2], A1

Εδώ φορτώνεται στον Α1 η τιμή fffffed9h η οποία βρίσκεται στη διεύθυνση που δείχνει ο Α10 συν δύο λέξεις. Η τιμή του Α10 παραμένει σταθερή.

1. LDW .D1 \*+A10[3], A1

Εδώ φορτώνεται στον Α1 η τιμή 4a7f78feh η οποία βρίσκεται στη διεύθυνση που δείχνει ο Α10 συν τρεις λέξεις. Η τιμή του Α10 παραμένει σταθερή.

1. LDW .D1 \*++A10[1], A1

Εδώ φορτώνεται στον Α1 η τιμή f9456f89h η οποία βρίσκεται στη διεύθυνση που δείχνει ο Α10 συν μια λέξη. Η τιμή του Α10 αυξάνεται κατά μία λέξη και είναι η 10ch.

1. LDW .D1 \*--A10[1], A1

Εδώ φορτώνεται στον Α1 η τιμή 3459f34dh η οποία βρίσκεται στη διεύθυνση που δείχνει ο Α10, μείον μια λέξη. Η τιμή του Α10 μειώνεται κατά μία λέξη και είναι η 104h.

1. LDB .D1 \*++A10[2], A1

Εδώ φορτώνεται στον Α1 η τιμή 2eh η οποία βρίσκεται στη διεύθυνση που δείχνει ο Α10 συν δύο bytes. Η τιμή του Α10 αυξάνεται κατά δύο bytes και είναι η 10ah.

1. LDΒ .D1 \*--A10[1], A1

Εδώ φορτώνεται στον Α1 η τιμή 34h η οποία βρίσκεται στη διεύθυνση που δείχνει ο Α10 μείον ένα byte. Η τιμή του Α10 μειώνεται κατά ένα byte και είναι η 107h.

1. LDW .D1 \*A10++[2], A1

Εδώ φορτώνεται στον Α1 η τιμή 5b2efe87h η οποία βρίσκεται στη διεύθυνση που δείχνει ο Α10. Η τιμή του Α10 αυξάνεται κατά δύο λέξεις και είναι η 110h.

1. LDW .D1 \*A10—[1], A1

Εδώ φορτώνεται στον Α1 η τιμή 5b2efe87h η οποία βρίσκεται στη διεύθυνση που δείχνει ο Α10. Η τιμή του Α10 μειώνεται κατά μία λέξη και είναι η 104h.

Διευκρίνηση: Θεωρήσαμε ότι κάθε εντολή εκτελείται ξεχωριστά και δεν αποτελούν ενιαίο πρόγραμμα.

Άσκηση 2.3

Σε αυτήν την άσκηση ζητείται να αποθηκεύσουμε την 32-bit σταθερά 53fe 23e4h στην θέση μνήμης 0000 0123h και να επιβεβαιώσουμε την ορθότητα του προγράμματος. Ο κώδικας δίνεται παρακάτω:

.def entry

.text

entry: MVKL .S1 0x00000124, A0

MVKH .S1 0x00000124, A0

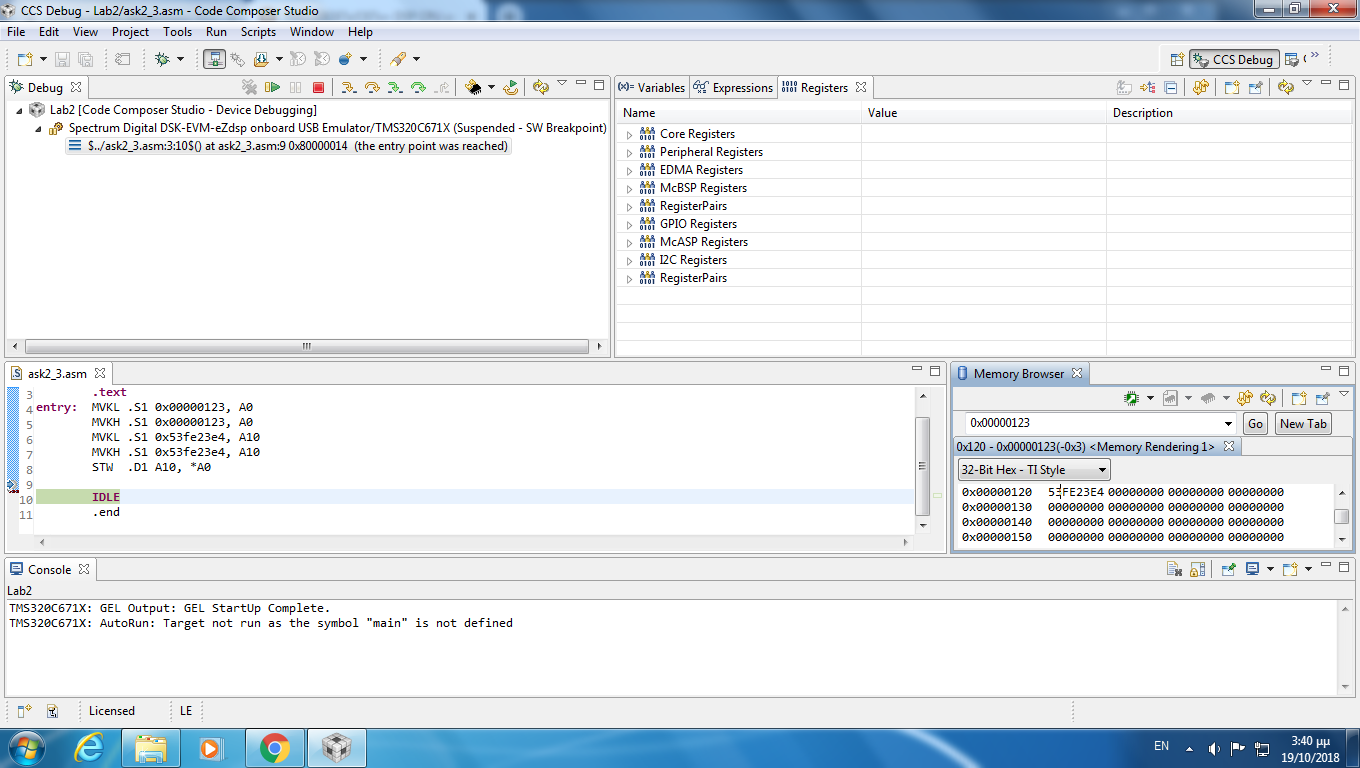
MVKL .S1 0x53fe23e4, A10

MVKH .S1 0x53fe23e4, A10

STW .D1 A10, \*A0

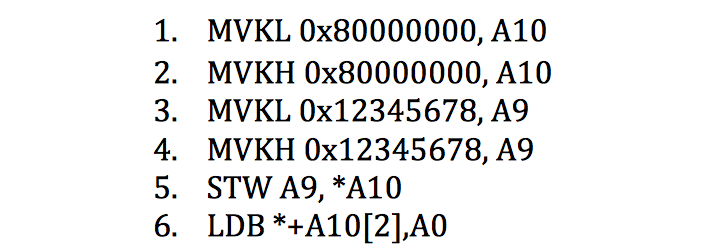
IDLE

.end

Όπως φαίνεται από την φωτογραφία και έπειτα από επιβεβαίωση των αποτελεσμάτων, τα αποτελέσματα είναι σωστά.

Άσκηση 2.4

Σε αυτήν την άσκηση ζητείται να υπολογίσουμε το περιεχόμενο Α0 μετά την εκτέλεση του κώδικα της εκφώνησης.



Αρχικά, φορτώνονται οι τιμές στους καταχωρητές Α10 και Α9, στη συνέχεια γίνεται η αποθήκευση της τιμής 0x12345678 στη θέση 0x80000000 της μνήμης. Για την περίπτωση little endian το byte που φορτώνεται είναι το 00, επειδή είναι το τρίτο byte ξεκινώντας από το LSB. Για την περίπτωση big endian το byte που φορτώνεται είναι το 00, επειδή είναι το τρίτο byte ξεκινώντας από το MSB.

Άσκηση 2.5

Σε αυτήν την άσκηση ζητείται να υπολογίσουμε την τιμή της παράστασης:

(0000 ef35h + 0000 33dch – 0000 1234h)\*0000 0007h.

Ο κώδικας είναι ο παρακάτω:

.def entry

.text

entry: MVKL .S1 0x0000ef35, A0

MVKH .S1 0x0000ef35, A0

MVK .S1 0x000033dc, A1

MVK .S1 0x00001234, A2

MVK .S1 0x00000007, A3

ADD .S1 A0, A1, A1

SUB .S1 A1, A2, A2

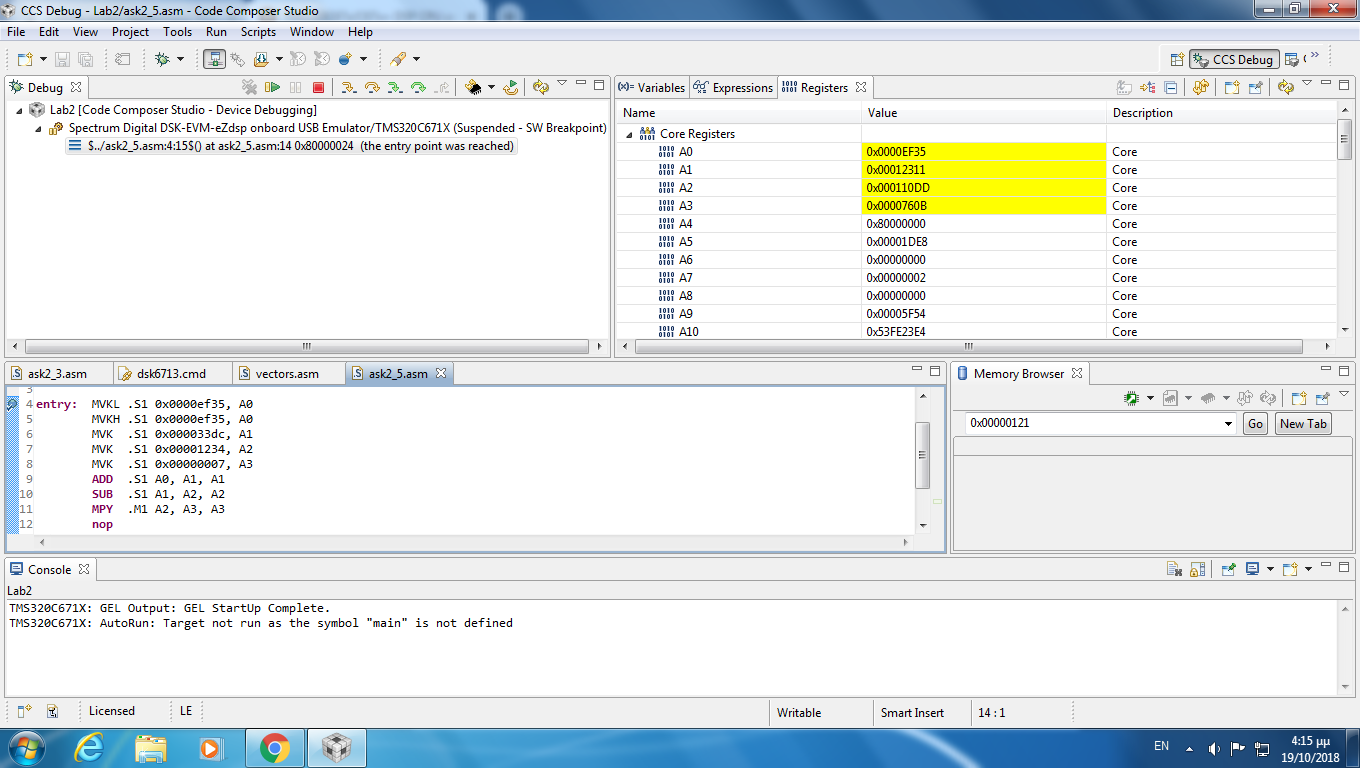
MPY .M1 A2, A3, A3

nop

IDLE

.end

Επιβεβαιώνοντας τα αποτελέσματα στο χαρτί και με την βοήθεια online calculator, είδαμε ότι τα αποτελέσματα είναι σωστά.



Άσκηση 2.6

Σε αυτήν την άσκηση μας ζητείται να υπολογιστεί το άθροισμα με εφαρμογή ενός απλού βρόχου. Ο κώδικας είναι ο παρακάτω:

.def entry

.text

entry: MVK .S2 0x1, B2

ZERO B1

MVK .S2 0x64, B0

loop: [B0] B .S2 loop

ADD .S2 B1, B0, B1

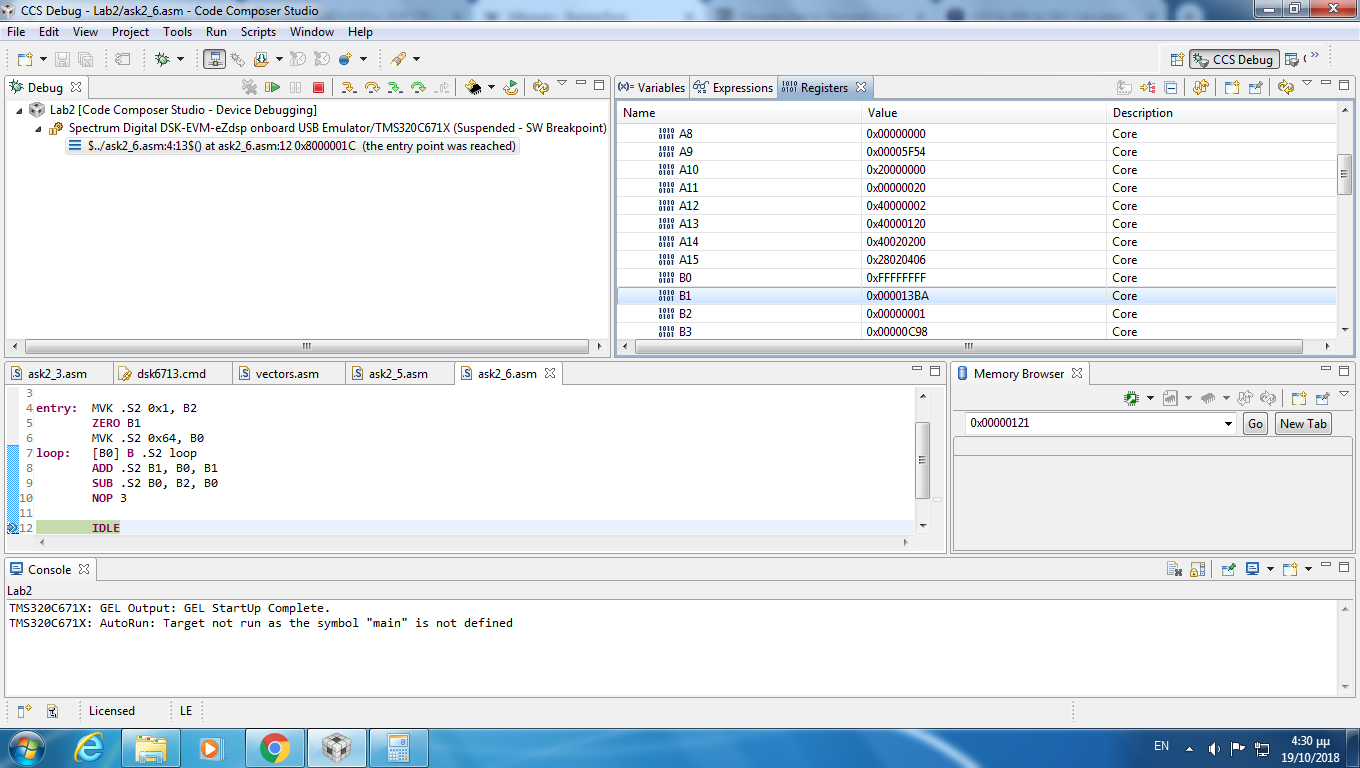
SUB .S2 B0, B2, B0

NOP 3

IDLE

.end

Επειδή η εντολή διακλάδωσης έχει delay slots πέντε κύκλους μπορούμε κατά τη διάρκεια αυτών των κύκλων να εκτελέσουμε τις εντολές πρόσθεσης και αφαίρεσης ώστε να «κερδίσουμε» κύκλους. Ελέγχοντας τα αποτελέσματα με χρήση του τύπου της αριθμητικής προόδου βρίσκουμε ότι είναι σωστά.



Δηλαδή το αποτέλεσμα θα είναι 5050 (13bah)

Άσκηση 2.7

Σε αυτήν την άσκηση μας ζητείται να αναθέσουμε λειτουργικές μονάδες στις εντολές που μας δίνονται. Με βάση το [[1]](#footnote-1) βλέπουμε τις λειτουργικές μονάδες που απαιτούνται για τις εντολές.

a) ADD .S1 A0, A1, A2

b) B .S1 A1

c) MVKL .S2 0x000023FE, B0

d) LDW .D1 \*A10, A3

Άσκηση 2.8

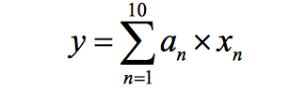
Σε αυτήν την άσκηση μας ζητείται να αναθέσουμε λειτουργικές μονάδες στις εντολές που μας δίνονται.

a) ADD .S2x B0, A1, B2

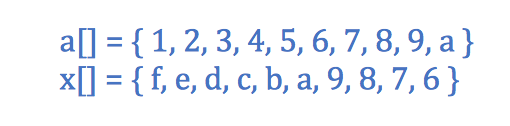
b) MPY .M1x A1, B2, A4

Άσκηση 2.9

Σε αυτήν την άσκηση μας ζητείται να υπολογίσουμε το εσωτερικό γινόμενο δυο διανυσμάτων μεγέθους 10 με βάση τον τύπο



Όπου τα διανύσματα είναι:



Ο κώδικας για την αποθήκευση των διανυσμάτων και τον υπολογισμό του εσωτερικού γινομένου είναι ο παρακάτω:

.def entry

.text

entry: MVK .S2 0x0120, B3 ; Address of a

MVK .S2 0xa, B1 ; Control value (10)

MVK .S2 0x1, B2 ; Counter

MVK .S2 0x1, B4 ; Increment value

loop: CMPEQ .L2 B2, B1, B0 ; if counter == 10

[!B0] B .S2 loop ; Prepare to branch in 5 cycles

STB .D2 B2, \*B3++[1] ; Store counter in a and

; increase address by 1 byte

ADD .S2 B2, B4, B2 ; Increase counter

NOP 3 ; Remaining cycles

; Branch!!!

MVK .S2 0x0120, B3 ; Set address to a[0]

MVK .S2 0x0130, B5 ; Address of x

MVK .S2 0x6, B1 ; Control value

MVKL .S2 0xf, B2

MVKH .S2 0xf, B2 ; Counter

; Decrement value = 1

loop2: CMPEQ .L2 B2, B1, B0 ; if counter == 6

[!B0] B .S2 loop2 ; Prepare to branch in 5 cycles

STB .D2 B2, \*B5++[1] ; Store counter in x and

; increase address by 1 byte

SUB .S2 B2, B4, B2 ; Decrease counter

NOP 3 ; Remaining cycles

; Branch!!!

MVK .S2 0x0130, B5 ; Set address to x[0]

ZERO B2 ; Counter

MVK .S2 0xa, B1 ; Control value

ZERO B8 ; y = 0

loop3: LDB \*B3++[1], B6 ; Load a (Will be completed

; after 4 additional cycles)

LDB \*B5++[1], B7 ; Load x (Will be completed

; after 4 additional cycles)

ADD .S2 B2, B4, B2 ; Increase counter

NOP 3 ; Wait for values a,x to be

; loaded

MPY .M2 B6, B7, B6 ; Multiply a\*x

nop ; Wait for a\*x to be calculated

ADD .S2 B8, B6, B8 ; y = y + a\*x

CMPEQ .L2 B2, B1, B0 ; if counter == 10

[!B0] B .S2 loop3 ; Prepare to branch in 5 cycles

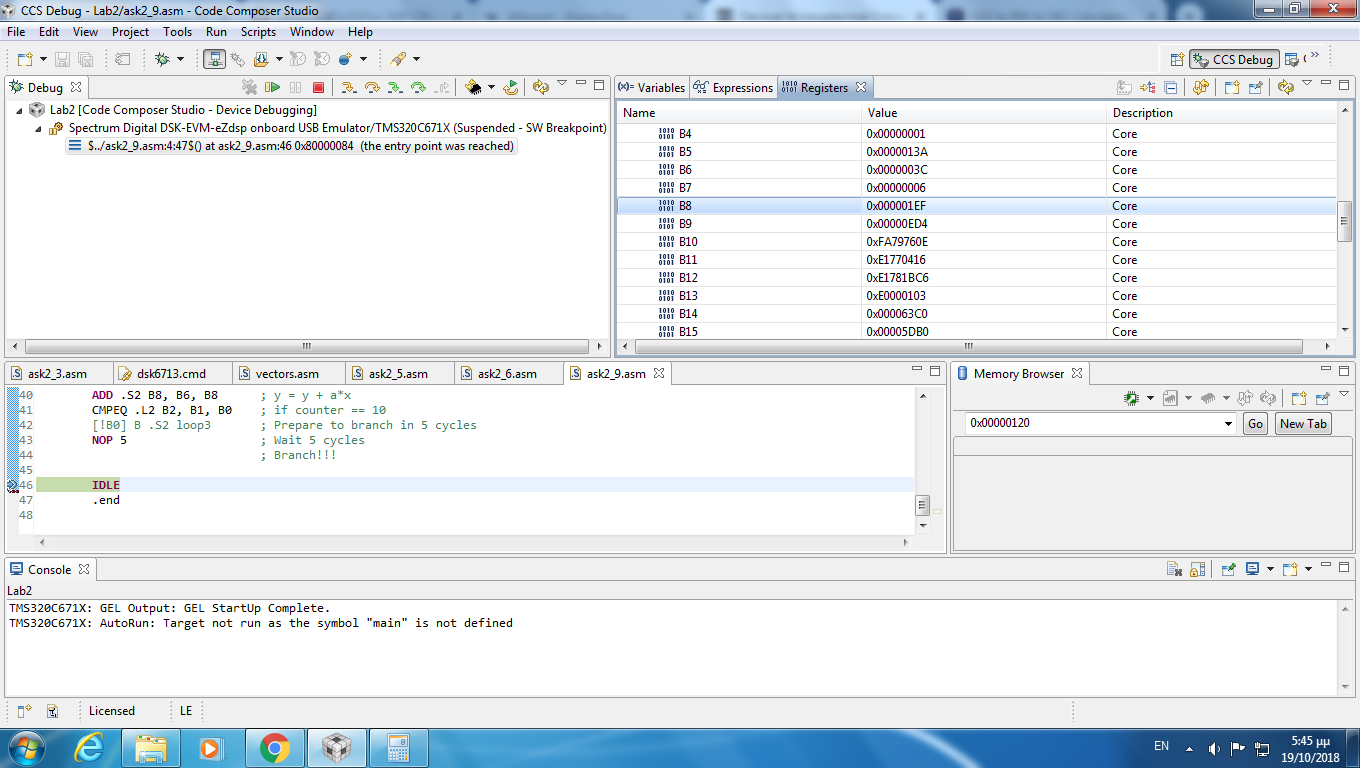
NOP 5 ; Wait 5 cycles

; Branch!!!

IDLE

.end

Στα σχόλια εξηγείται η λογική που ακολουθεί ο κώδικας. Με το χέρι υπολογίστηκε ότι το εσωτερικό γινόμενο θα πρέπει να κάνει 495 (01efh)



Άσκηση 3.6

Σε αυτό το ερώτημα, είδαμε πώς να κάνουμε ανάγνωση και εγγραφή στην μνήμη Ι/Ο ώστε να ελέγξουμε μέσω διακοπτών τα περιφερειακά LEDs. Το τελευταίο LED πρέπει να παραμένει πάντα κλειστό. Ο κώδικας είναι ο παρακάτω:

.def entry

**.text**

**entry:**  MVKL .S2 0x90080000, B0

MVKH .S2 0x90080000, B0 ; Load the address

MVK .S2 0x4, B2 ; Load bit shifts

MVK .S2 0x7, B4

**loop:** LDW .D2 \*B0, B1 ; Load the bit set (switch, led states)

**NOP** 4 ; Wait to load

[B0] B .S2 loop ; Prepare to branch

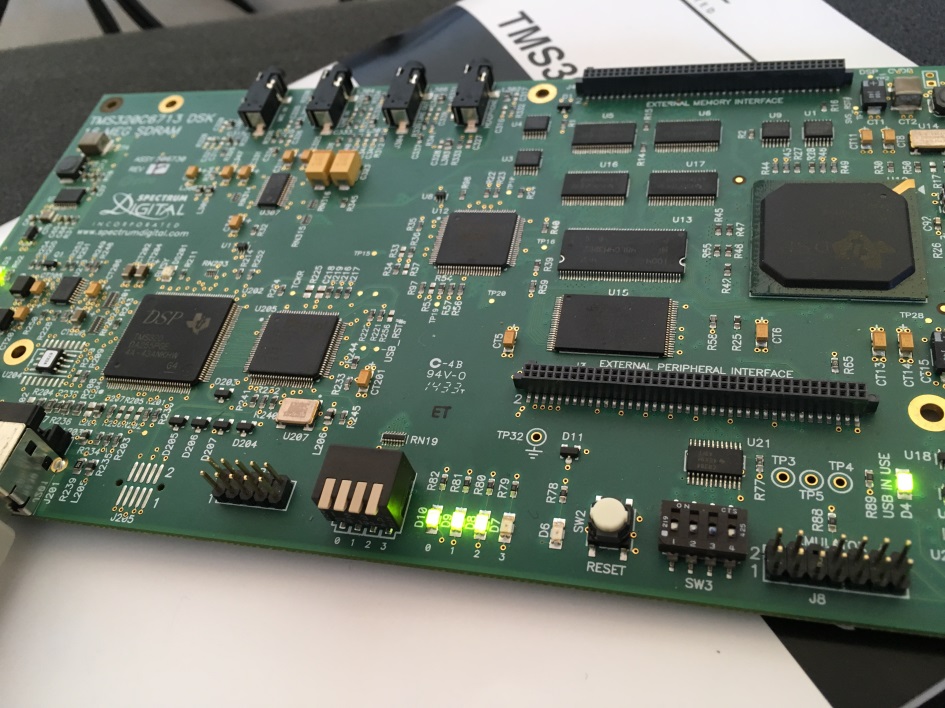
SHRU .S2 B1, B2, B3 ; Shift right 4 times

**AND** .S2 B3, B4, B5 ; Bitwise AND to turn off LED no4

STB .D2 B5, \*B0 ; Store states of switches to states of LEDs

**NOP** 3 ; Wait to branch

Αρχικά διαβάζουμε από την συγκεκριμένη διεύθυνση Ι/Ο τις τιμές μίας λέξης 32 bit. Έπειτα με βάση το little endian μοντέλο, πρέπει να γίνει δεξιά μετατόπιση κατά 4, ώστε οι καταστάσεις των διακοπτών να γίνουν οι καταστάσεις των LEDs. Ακόμη κρατάμε το τελευταίο LED μόνιμα σβηστό μέσω της πράξης AND (little endian στο 4ο LED). Το αποτέλεσμα φαίνεται στην παρακάτω εικόνα:



1. [TMS320C67x/C67x+ DSP CPU and Instruction Set](https://eclass.upatras.gr/modules/document/file.php/EE873/%CE%91%CE%BA%CE%B1%CE%B4%CE%B7%CE%BC%CE%B1%CF%8A%CE%BA%CF%8C%20%CE%88%CF%84%CE%BF%CF%82%202018-2019/%CE%86%CF%83%CE%BA%CE%B7%CF%83%CE%B7%200%20-%20%CE%95%CE%B9%CF%83%CE%B1%CE%B3%CF%89%CE%B3%CE%AE%20%CF%83%CF%84%CE%BF%20%CE%B5%CF%81%CE%B3%CE%B1%CF%83%CF%84%CE%AE%CF%81%CE%B9%CE%BF/SPRU733A%20-%20C6713%20Instruction%20Set.pdf) [↑](#footnote-ref-1)