Τίτλος Ασκησης:	Εισαγωγή στην Assembly για επεξεργαστές x86
Εργαστήριο:	1
Απαραίτητα Εργαλεία:	Visual Studio Professional 2008 (προτεινόμενο) /2010/2012 (και εκδόσεις express)
Απαιτούμενες Γνώσεις:	C/C++, Αρχιτεκτονική επεξεργαστών, Ψηφιακή σχεδίαση
Στόχοι:	1) Εξοικείωση και εισαγωγή στις βασικές έννοιες της αρχιτεκτονικής επεξεργαστών x86 2) Εξοικείωση και εισαγωγή στις βασικές έννοιες της συμβολικής γλώσσας Assembly 3) Εξοικείωση με το περιβάλλον του Visual Studio
Διορία Παράδοσης:	Εντός της διδακτική ώρας του επόμενου εργαστηρίου
Παραδοτέα:	1) Επιδειξη λειτουργίας και Πηγαία Αρχεία (5/10) 2) Γραπτή αναφορά (στο χαρτί) των ενεργειών που έγιναν (5/10)
Διδάσκων:	Γρηγόρης Δημητρουλάκος

Στόχος εργαστηριακής άσκησης : Στην παρούσα άσκηση ο φοιτητής καλείται να εγκαταστήσει και παραμετροποιήσει τα απαραίτητα εργαλεία προκειμένου να είναι σε θέση να εκτελεί προγράμματα σε συμβολική γλώσσα Assembly για επεξεργαστές τεχνολογίας x86. Παράλληλα καλείται να γράψει ένα αρχικό στοιχειώδες assembly πρόγραμμα, χρησιμοποιώντας μια (έτοιμη) βασική βιβλιοθήκη assembly διαδικασιών, προκειμένου το πρόγραμμα να μπορεί να επικοινωνεί με τις βασικές συσκευές εισόδου – εξόδου του συστήματος. Επίσης ο φοιτητής καλείται να κάνει έλεγχο εκτέλεσης του προγράμματος (debugging), παρακολουθώντας την μεταβολή των παραμέτρων (registers) της CPU καθώς και των σχετικών περιεχομένων της μνήμης του συστήματος (προγράμματος).

Άσκηση 1: Δημιουργία ενός απλού προγράμματος σε γλώσσα assembly (χωρίς συμπερίληψη βοηθητικού αρχείου δηλώσεων) που προσθέτει τον αριθμό 50000h στον 10000h και στη συνέχεια θα αφαιρεί τον αριθμό 20000h, εμφανίζοντας το τελικό περιεχόμενο των registers της CPU στην οθόνη (console) του συστήματος.

α) Εγκατάσταση εργαλείου Visual Studio

b) Εγκατάσταση assembly βοηθητικών βιβλιοθηκών

c) Παραμετροποίηση Visual Studio

d) Δημιουργία νέου project

e) Δημιουργία πηγαίου αρχείου assembly κώδικα με όνομα **Ergasia_1_a.asm** με χρήση των βασικών εντολών (mov, add, sub, call). Αρχικά η τιμή 10000h θα φορτώνεται απευθείας (immediate) στον 32-bit register EAX, στη συνέχεια θα προστίθεται (στον EAX) η απευθείας τιμή 50000h και κατόπιν θα αφαιρείται (από τον EAX) η απευθείας τιμή 30000h. Τέλος πρέπει να πραγματοποιείται κλήση της procedure DumpRegs (περιέχεται στη βοηθητική βιβλιοθήκη) για την εμφάνιση των τιμών των CPU registers στην οθόνη. Η βασική δομή του προγράμματος έχει ως εξής :

```
TITLE  Add and Subtract            (Ergasia_1_a.asm)
; This program adds and subtracts 32-bit integers.
        .386
        .model flat, stdcall
        .stack 4096
        ExitProcess PROTO,  dwExitCode:DWORD
        DumpRegs PROTO
```

```
.code
main PROC
                {εδώ θα εισαχθούν οι εντολές για την υλοποίηση της άσκησης}
    INVOKE ExitProcess,0
main ENDP
END main
```

f) Μεταγλώττιση (assembling) και διασύνδεση (linking) προγράμματος

g) Εκτέλεση (step by step) προγράμματος σε κατάσταση αποσφαλμάτωσης (debugging) με παράλληλη εμφάνιση των τιμών των register της CPU

Άσκηση 2: Μετατροπή του ανωτέρω προγράμματος με συμπερίληψη του βοηθητικού αρχείου δηλώσεων irvine32.inc της βοηθητικής βιβλιοθήκης :

α) Αντικατάσταση των αρχικών δηλώσεων (έντονη γραφή προηγούμενης άσκησης) με την συμπερίληψη του αρχείου επικεφαλίδων Irvine32.inc της βοηθητικής βιβλιοθήκης. Τελική μορφή κώδικα :

```
TITLE  Add and Subtract          (Ergasia_1_b.asm)
; This program adds and subtracts 32-bit integers.
INCLUDE Irvine32.inc
.code
main PROC
        {εντολές για την υλοποίηση της άσκησης}
        exit
main ENDP
END main
```

b) Μεταγλώττιση (assembling) και διασύνδεση (linking) προγράμματος

c) Εκτέλεση προγράμματος

Υποδείξεις :

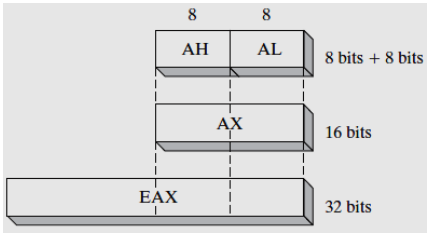
1) **Visual Studio** : οδηγίες για την εγκατάσταση και παραμετροποίηση του Visual Studio και των βιβλιοθηκών καθώς και την παραμετροποίηση του Assembler, μπορείτε να βρείτε στο υποστηρικτικό υλικό που συνοδεύει την άσκηση
2) **Assembler** : είναι ο μεταγλωττιστής της συμβολικής γλώσσας assembly (αντίστοιχος του C compiler για c/c++ προγράμματα) ο οποίος και παράγει τον object (*.obj) κώδικα του προγράμματος (στη συνέχεια ο linker αναλαμβάνει την δημιουργία του εκτελέσιμου αρχείου *.exe από την διασύνδεση των object και library files του project
3) **MASM** : ο Microsoft Assembler που εμπεριέχεται στο visual studio (δείτε το υποστηρικτικό υλικό για την ενεργοποίηση του – στο πρότυπο project είναι είδη ενεργοποιημένος) προκειμένου να μεταγλωττίζονται assembly προγράμματα σε γλώσσα μηχανής (machine language)
4) **Κύρια βήματα** : ξεκινήστε ανοίγοντας το Visual Studio και δημιουργώντας ένα νέο project

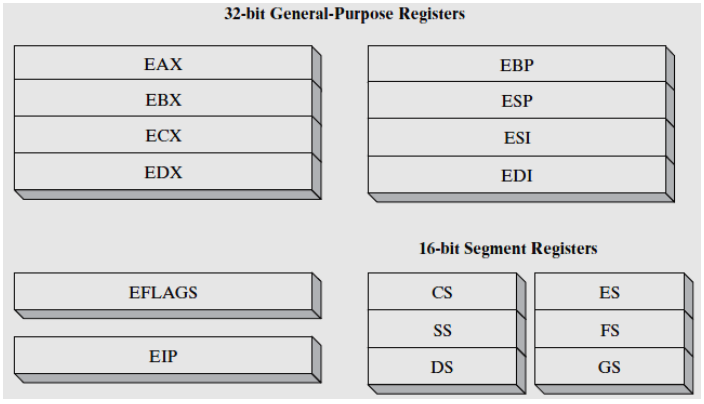
(σύμφωνα με τις οδηγίες παραμετροποίησης του υποστηρικτικού υλικού της άσκησης), στη συνέχεια δημιουργήστε ένα νέο αρχείο πηγαίου κώδικα με όνομα **Ergasia_1_a.asm** και γράψτε μέσα σε αυτό το πρόγραμμά σας.

5) **Βασικές εντολές assembly :** η γενική σύνταξη των απαιτούμενων βασικών assembly εντολών για την άσκηση έχουν ως εξής :

MOV *dest,source*	ADD *dest,source*	SUB *dest,source*	CALL *procedure_name*
MOV *reg,reg*	*reg,reg*	*reg,reg*	
MOV *mem,reg*	*mem,reg*	*mem,reg*	
MOV *reg,mem*	*reg,mem*	*reg,mem*	
MOV *mem,imm*	*mem,imm*	*mem,imm*	
MOV *reg,imm*	*reg,imm*	*reg,imm*	

6) **Αναφορές reg/imm**: όπου reg χρησιμοποιείστε τον 32-bit καταχωρητή γενικού σκοπού με όνομα **EAX**. Οι απευθείας (imm) αναφορά αριθμητικών τιμών σε ένα πρόγραμμα assembly ακολουθεί εξορισμού το δεκαδικό σύστημα αρίθμησης εκτός αν ο αριθμός συνοδεύεται από χαρακτήρα προσδιορισμού βάσης (π.χ. 1A**h** = 26d) . Παρουσίαση των βασικών αρχών της αρχιτεκτονικής σχεδίασης επεξεργαστών και της γλώσσας assembly (π.χ. παράμετροι περιβάλλοντος, ονόματα καταχωρητών, αναπαράσταση αριθμητικών τιμών, κ.λ.π.) για x86 CPU, μπορείτε να βρείτε στο υποστηρικτικό υλικό που συνοδεύει την άσκηση
7) **Βασικοί x86 registers / EAX register** : σχηματική αναπαράσταση καταχωρητών γενικού σκοπού της CPU





8) **Βιβλιοθήκη Irvine** : τα σχετικά αρχεία περιέχονται στο υποστηρικτικό υλικό που συνοδεύει την άσκηση καθώς και οδηγίες για την εγκατάσταση τους
9) **Complex Instruction Set Computer (CISC) :** Η σχεδίαση CISC εισήχθη αρχικά στον επεξεργαστή Intel 8086 παρέχοντας ένα μεγάλο σύνολο εντολών περιλαμβάνοντας πολλαπλούς τρόπους διευθυνσιοδότησης, shifting, αριθμητικές και λογικές πράξεις, data movement, κλπ.
10) **32-bit Protected Mode** : είναι η εξορισμού κατάσταση λειτουργίας του επεξεργαστή στην οποία όλες οι εντολές είναι διαθέσιμες. Στα προγράμματα εκχωρούνται (από το OS) ξεχωριστές περιοχές-τμήματα μνήμης (**segments**) και ο επεξεργαστής αποτρέπει στα προγράμματα να αναφέρονται σε μνήμη εκτός της εκχωρημένης περιοχής τους (segment)
11) Τόσο τα ονόματα των εντολών όσο και τα ονόματα των καταχωρητών, αντιμετωπίζονται από τον MASM ως case insensitive και μπορούν γραφούν με μικρούς ή κεφαλαίους χαρακτήρες. Κάθε γραμμή κώδικα μπορεί να περιέχει μία εντολή assembly. Σχόλια στον κώδικα μπορούν να προστεθούν μετά τον χαρακτήρα ';' και μέχρι το τέλος της τρέχουσας γραμμής.
12) **Μελέτη** : λόγω της ιδιαίτερα ασυνήθιστης φύσης της γλώσσας assembly σε σχέση με τις γλώσσες υψηλού επιπέδου, κρίνεται σκόπιμο να μελετήσετε αναλυτικά το υποστηρικτικό υλικό που συνοδεύει την άσκηση προκειμένου να κατανοήσετε τον τρόπο λειτουργίας της (μην επικεντρωθείτε επί του παρόντος με θέματα real-address mode, αριθμών κινητής υποδιαστολής, strings, τρόπων διευθυνσιοδότησης και εντολών πέραν των εμπλεκομένων στη

άσκηση). Για την απάντηση των ερωτημάτων μπορείτε να πειραματισθείτε με διαφορετικές αριθμητικές τιμές στο πρόγραμμα σας, παρακολουθώντας τα περιεχόμενα των registers της CPU σε κατάσταση debugging

13) Το εκτελέσιμο αρχείο **Project_Add_Sub.exe** (περιλαμβάνεται στο υποστηρικτικό υλικό της άσκησης) παρουσιάζει το τελικό αποτέλεσμα εξόδου που αναμένεται από την άσκηση

Περιεχόμενα Γραπτής Αναφοράς

1) Κώδικας αρχείου **Ergasia_1_a.asm** του βήματος (1.e)
2) Screen shot του visual studio σε κατάσταση debugging, όπου να παρουσιάζονται το πρόγραμμα και τα περιεχόμενα των register (ακριβώς πριν την εκτέλεση της εντολής τερματισμού του προγράμματος), του βήματος (1.g)
3) Κώδικας αρχείου **Ergasia_1_b.asm** του βήματος (2.a)
4) Σύντομη (πολύ γενική) περιγραφή των αρχικών δηλώσεων **.386, .model flat, stdcall** και **.stack 4096**
5) Γιατί νομίζετε ότι απαιτείται η δήλωση **PROTO** των διαδικασιών **ExitProcess** και **DumpRegs** που καλεί το πρόγραμμα στην $1^{η}$ άσκηση?
6) Τι είναι το αρχείο irvine32.inc και σε πιο στάδιο (μεταγλώττισης ή διασύνδεσης) της ανάπτυξης εφαρμογής συμμετέχει ?
7) Που βρίσκεται ο κώδικας των καλούμενων διαδικασιών **ExitProcess** και **DumpRegs** και πως (με πιες ρυθμίσεις) το Visual Studio επιτυγχάνει τη χρησιμοποίηση τους στο τρέχον project ?
8) Με ποιο τρόπο η CPU υλοποιεί την εντολή αφαίρεσης SUB σε φυσικό επίπεδο ?
9) Πως αναπαρίστανται οι ακέραιοι αρνητικοί αριθμοί στη CPU, δώστε ένα παράδειγμα μετατροπής των αρνητικών τιμών -123d και -2ABh σε μορφή κατάλληλη για την CPU (32-bit μέγεθος αναπαράστασης)
10) Η πράξη 10d+(-5d) πώς υλοποιείται από τη CPU ? με πρόσθεση ADD ή αφαίρεση SUB και γιατί ?