Τίτλος Ασκησης:	Αποτίμηση απλών αριθμητικών εκφράσεων σε γλώσσα Assembly x86
Εργαστήριο:	2
Απαραίτητα Εργαλεία:	Visual Studio Professional 2008 (προτεινόμενο) /2010/2012 (και εκδόσεις express)
Απαιτούμενες Γνώσεις:	Εισαγωγή στη Assembly, C/C++, Αρχιτεκτονική επεξεργαστών, Ψηφιακή σχεδίαση
Στόχοι:	1) Ορισμός συμβολικών ονομάτων στην Assembly 2) Βασικοί τύποι δεδομένων, και ορισμός μεταβλητών στην Assembly 3) Αναπαράσταση ακεραίων αριθμητικών δεδομένων και τρόπος αποθήκευσης στην μνήμη 4) Χρησιμοποίηση των βασικών εντολών μετακίνησης δεδομένων (MOV) και πράξεων (ADD, SUB) για την αποτίμηση απλών αριθμητικών εκφράσεων σε Assembly x86 5) Εξαγωγή του Listing File
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 γράψει ένα assembly πρόγραμμα για την αποτίμηση απλών αριθμητικών εκφράσεων ακέραιων αριθμών. Παράλληλα ο φοιτητής εξοικειώνεται με τους βασικούς τύπους δεδομένων που υποστηρίζει η Assembly, τον ορισμό συμβολικών ονομάτων και σταθερών, τον ορισμό και αρχικοποίηση μεταβλητών, την χρήση των βασικών εντολών MOV, ADD, SUB, την άμεση αναφορά στη μνήμη (direct memory access), τον τρόπο αποθήκευσης των δεδομένων στη μνήμη, καθώς και την εξαγωγή του αρχείου Listing File.

Άσκηση 1: Δημιουργία ενός προγράμματος σε γλώσσα assembly το οποίο να υπολογίζει το αποτέλεσμα της αριθμητικής έκφρασης **w = (x + y) – (50d + x + y + z - p)**, όπου α) τα w, x, y, z είναι συμβολικά ονόματα ακέραιων προσημασμένων μεταβλητών 32-bit, αποθηκευμένα στη μνήμη με αρχικές τιμές που ορίζονται κατά την δήλωσή τους και β) το p είναι συμβολικό όνομα σταθεράς με τιμή p=8d. Το αποτέλεσμα της έκφρασης θα αποθηκεύεται στη μεταβλητή w και θα εμφανίζεται στην οθόνη (console) του συστήματος σε δεκαδική προσημασμένη μορφή.

α) Δημιουργία νέου project για προγράμματα assembly ή χρήση προτύπου από προηγούμενη άσκηση

b) Δημιουργία πηγαίου αρχείου assembly κώδικα με όνομα **Ergasia_2_a.asm,** όπου με χρήση των βασικών εντολών (mov, add, sub, call) θα υπολογίζεται και θα εμφανίζεται το αποτέλεσμα της ζητούμενης έκφρασης. Αρχικά, θα δηλώνεται η σταθερά με συμβολικό όνομα p και τιμή p=8d με χρήση της οδηγίας EQU

c) Στη συνέχεια, στο data segment του προγράμματος, θα δηλώνονται οι μεταβλητές με συμβολικά ονόματα x, y, z, w τύπου προσημασμένου ακεραίου, μεγέθους 32-bit, με αρχικοποιημένες τιμές όπως παρακάτω :

x = 16d, y = 154d , z = -17d, w = *χωρίς τιμή*

d) Κατόπιν θα μεταφέρονται οι τιμές x,y σε καταχωρητή/ες της CPU και θα υπολογίζεται το πρώτο μέρος της έκφρασης (x + y) με χρήση των εντολών MOV, ADD. Στη συνέχεια με αντίστοιχο τρόπο θα υπολογίζεται το δεύτερο μέρος της έκφρασης (50d + x + y + z - p) με χρήση των εντολών MOV, ADD, SUB. Το τελικό αποτέλεσμα θα υπολογίζεται και αποθηκεύεται στην μεταβλητή w με χρήση των εντολών MOV, SUB.

e) Τέλος το αποτέλεσμα (της μεταβλητής w) θα πρέπει να μεταφερθεί στον EAX register και στη συνέχεια θα πραγματοποιείται κλήση της procedure με όνομα WriteInt (περιέχεται στη βοηθητική βιβλιοθήκη irvine32) για την εμφάνιση της τιμής (αποτελέσματος στον EAX) στην οθόνη σε δεκαδική προσημασμένη μορφή. Επιπρόσθετα μπορεί να κληθεί και η procedure με όνομα Crlf (περιέχεται στη βοηθητική βιβλιοθήκη irvine32) για την αλλαγή γραμμής στην οθόνη

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

g) Για λόγους ομοιομορφίας ακολουθείστε το παρακάτω πρότυπο κώδικα:

```
TITLE  Expression calculator       (Ergasia_2_a.asm)
; This program calculates a simple arithmetical expression
INCLUDE Irvine32.inc
        {δήλωση σταθερών συμβολικών ονομάτων – compile time data}
.data
        {δήλωση συμβολικών ονομάτων μεταβλητών – run time data}
.code
main PROC
        {εντολές για την υλοποίηση της άσκησης}
        exit
main ENDP
END main
```

Άσκηση 2: Μετατροπή του ανωτέρω προγράμματος (με όνομα **Ergasia_2_b.asm**) ώστε να υλοποιηθεί με λιγότερες εντολές, εκμεταλλευόμενοι την επανεμφάνιση συγκεκριμένων υπο-εκφράσεων

α) Εντοπισμός επαναλαμβανόμενων υπο-εκφράσεων στην υπό αποτίμηση έκφραση και αποθήκευση ενδιάμεσου αποτελέσματος σε ξεχωριστό register (π.χ. EBX)

b) Αναδιατύπωση των εντολών υπολογισμού της έκφρασής ώστε να επαναχρησιμοποιείται το ενδιάμεσο αποτέλεσμα του προηγούμενου βήματος (αντί να επανα-υπολογίζεται η ίδια υπο-έκφραση)

c) Μεταγλώττιση (assembling), διασύνδεση (linking) και εκτέλεση του προγράμματος

Υποδείξεις :

1) **Symbol names** : τα συμβολικά ονόματα στην assembly ακολουθούν συγκεκριμένους κανόνες σύνταξης (δεν μπορούν είναι δεσμευμένες λέξεις, να ξεκινούν από ψηφίο, κλπ) και χρησιμοποιούνται για δηλώσεις συμβολικών ονομάτων σταθερών, μεταβλητών, δομών, procedures, macros, labels, κλπ. Η ισχύ τους αφορά μόνο τον assembler και το χρόνο μεταγλώττισης του προγράμματος. Στο τελικό εκτελέσιμο κώδικα τα symbol names αντικαθίστανται από συγκεκριμένες σχετικές διευθύνσεις (offsets) ή άμεσες τιμές (immediate values)
2) **EQU directive** : οδηγία προς τον assembler για την δήλωση συμβολικών ονομάτων <u>σταθερών</u> που χρησιμοποιούνται και υπολογίζονται από τον προεπεξεργαστή του assembler μόνο κατά την μεταγλώττιση του προγράμματος (compile time data). Παραδείγματα δηλώσεων συμβολικών ονομάτων σταθερών :

```
sym1 EQU 3d+5d          ; expression assignment, sym1 = 8d
sym2 EQU <3d+5d >       ; text assignment, sym2 = "3d+5d"
```

3) **Data segment** : το τμήμα του πηγαίου κώδικα με επικεφαλίδα **.data**, εντός του οποίου δηλώνονται τα συμβολικά ονόματα των μεταβλητών των δεδομένων που χρησιμοποιεί το πρόγραμμα. Ταυτόχρονα μπορούμε να αποδώσουμε και αρχικές τιμές σε όποια δεδομένα επιθυμούμε. Τα δεδομένα αυτά αποθηκεύονται στη μνήμη του προγράμματος σε σταθερές (σχετικές) διευθύνσεις (offset) και μπορούν να μεταβάλλονται κατά το χρόνο εκτέλεσης του προγράμματος (run time data).
4) **Code Segment** : το τμήμα του πηγαίου κώδικα με επικεφαλίδα **.code**, εντός του οποίου περιέχονται οι δηλώσεις – εντολές του προγράμματος
5) **Variables** : για τον assembler μια μεταβλητή είναι το συμβολικό όνομα μιας σχετικής θέσης μνήμης (offset) εντός του τμήματος δεδομένων, που υποδεικνύει το πρώτο byte, όπου είναι αποθηκευμένα τα σχετικά δεδομένα της μεταβλητής (ανάλογα με τον τύπο των δεδομένων)
6) **Little Endian Order** : οι x86 επεξεργαστές αποθηκεύουν (save) και ανακτούν (load) τα δεδομένα από τη μνήμη χρησιμοποιώντας την little endian διάταξη (low to high). Το λιγότερο σημαντικό (LS) byte αποθηκεύεται πρώτο στη μνήμη που έχει δεσμευθεί για τον συγκεκριμένο τύπο δεδομένου, τα υπόλοιπα bytes αποθηκεύονται με την αντίστροφη σειρά στις αμέσως επόμενες θέσεις της μνήμης. Παράδειγμα αναπαράστασης μνήμης του 32-bit αριθμού 12345678h →

0000:	78
0001:	56
0002:	34
0003:	12

7) **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*	

Ανάλυση σημειογραφίας των operand των εντολών:

Operand	Description
reg8	8-bit general-purpose register: AH, AL, BH, BL, CH, CL, DH, DL
reg16	16-bit general-purpose register: AX, BX, CX, DX, SI, DI, SP, BP
reg32	32-bit general-purpose register: EAX, EBX, ECX, EDX, ESI, EDI, ESP, EBP
reg	Any general-purpose register
sreg	16-bit segment register: CS, DS, SS, ES, FS, GS
imm	8-, 16-, or 32-bit immediate value
imm8	8-bit immediate byte value
imm16	16-bit immediate word value
imm32	32-bit immediate doubleword value
reg/mem8	8-bit operand, which can be an 8-bit general register or memory byte
reg/mem16	16-bit operand, which can be a 16-bit general register or memory word
reg/mem32	32-bit operand, which can be a 32-bit general register or memory doubleword
mem	An 8-, 16-, or 32-bit memory operand

8) Αναφορές **reg/imm**: όπου reg χρησιμοποιείστε όποιους από τους 32-bit καταχωρητές γενικού σκοπού με ονόματα **EAX, EBX, ECX, EDX**. 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).
9) Αναφορές **mem** : όπου mem χρησιμοποιείστε τα συμβολικά ονόματα των σταθερών ή/και μεταβλητών του προγράμματος. Η χρήση συμβολικού ονόματος σταθεράς ισοδυναμεί με μια άμεση (**immediate**) απόδοση τιμής (πχ. ADD eax, p ←→ ADD eax, 8d). Η χρήση συμβολικού ονόματος μεταβλητής αποτελεί μια άμεση αναφορά στη μνήμη (**direct memory access**), εφόσον για τον assembler το συμβολικό όνομα μιας μεταβλητής είναι η σχετική διεύθυνση (offset) στο τμήμα δεδομένων όπου και είναι αποθηκευμένα τα δεδομένα της (πχ. ADD eax, x ←→ ADD eax, [00058000h] όπου 00058000h είναι η σχετική διεύθυνση – offset στο τμήμα δεδομένων όπου και είναι αποθηκευμένη η τιμή x=16d). Αναλυτικές πληροφορίες και οδηγίες σχετικά με τις εντολές της άσκησης μπορείτε να βρείτε στο υποστηρικτικό υλικό που συνοδεύει την άσκηση
10) **Βασικοί τύποι δεδομένων** : ο MASM υποστηρίζει τους παρακάτω βασικούς τύπους δεδομένων :

Type	Usage
BYTE	8-bit unsigned integer. B stands for byte
SBYTE	8-bit signed integer. S stands for signed
WORD	16-bit unsigned integer (can also be a Near pointer in real-address mode)
SWORD	16-bit signed integer
DWORD	32-bit unsigned integer (can also be a Near pointer in protected mode). D stands for double
SDWORD	32-bit signed integer. SD stands for signed double
FWORD	48-bit integer (Far pointer in protected mode)
QWORD	64-bit integer. Q stands for quad
TBYTE	80-bit (10-byte) integer. T stands for Ten-byte
REAL4	32-bit (4-byte) IEEE short real
REAL8	64-bit (8-byte) IEEE long real
REAL10	80-bit (10-byte) IEEE extended real

(αναλυτικές πληροφορίες για τον κάθε τύπο δεδομένων μπορείτε να βρείτε στο υποστηρικτικό υλικό που συνοδεύει την άσκηση)

Παραδείγματα δηλώσεων συμβολικών ονομάτων ακέραιων μεταβλητών :

```
.data
var1 WORD ?          ; 16-bit unsigned integer, no initial value
var2 SBYTE -5d       ; 8-bit signed integer, initial value = -5d
```

11) **WriteInt** procedure: διαδικασία της βοηθητικής βιβλιοθήκης irvine32, η οποία εμφανίζει την τρέχουσα τιμή του EAX register στην οθόνη (console window) σε δεκαδική προσημασμένη μορφή (πχ. αν EAX=FFFFFFFDh τότε εμφανίζει την δεκαδική τιμή -3)
12) **Crlf** procedure : διαδικασία της βοηθητικής βιβλιοθήκης irvine32, η οποία μεταφέρει τον κέρσορα της οθόνης (console window) στην αρχή της επόμενης γραμμής
13) **Listing File** : πληροφοριακό αρχείο που παράγεται από τον assembler, κατά την μεταγλώττιση, και περιέχει (εκτός των άλλων) ένα αντίγραφο του πηγαίου κώδικα (source code) του προγράμματος assembly, με αριθμούς γραμμών, μετατοπίσεις διευθύνσεων (offset address), μεταφρασμένους κώδικες μηχανής (machine code) και ένα πίνακα συμβόλων (symbol table). Εντοπίζεται μέσα στον φάκελο του project με όνομα source_file_name.lst
14) **Debugging memory** : στο visual studio και σε κατάσταση debugging, με ανοιχτό το παράθυρο Memory1, μπορείτε στο πεδίο Address να εισάγεται (αντί την τιμή μίας διεύθυνσης) την τιμή &x, έτσι ώστε τα περιεχόμενα της μνήμης του παραθύρου να προσαρμοσθούν με βάση το offset (σχετική διεύθυνση) του πρώτου byte της μεταβλητής x. Η δυνατότητα αυτή είναι αρκετά χρήσιμη ώστε να παρακολουθείτε τα περιεχόμενα και τον τρόπο αποθήκευσης των δεδομένων, μιας μεταβλητής (με όνομα έστω x) του προγράμματος, στη μνήμη σε πραγματικό (runtime) χρόνο
15) Το εκτελέσιμο αρχείο **Project_Expression_Simple.exe** (περιλαμβάνεται στο υποστηρικτικό υλικό της άσκησης) παρουσιάζει το τελικό αποτέλεσμα εξόδου που αναμένεται από την άσκηση

Περιεχόμενα Γραπτής Αναφοράς

1) Κώδικας αρχείου **Ergasia_2_a.asm** της άσκησης 1
2) Screen shot του visual studio σε κατάσταση debugging, όπου να παρουσιάζονται το πρόγραμμα και τα περιεχόμενα των register καθώς και της data segment memory όπου είναι αποθηκευμένες οι μεταβλητές του προγράμματος (ακριβώς πριν την εκτέλεση της εντολής τερματισμού του προγράμματος)
3) Κώδικας αρχείου **Ergasia_2_b.asm** της άσκησης 2
4) Αντίστοιχο Screen shot του visual studio με το (2)
5) Το περιεχόμενο του αρχείου Listing File που παράχθηκε κατά την μεταγλώττιση της άσκησης 2
6) Ποια είναι τα μεγέθη (σε bytes) των data και code segment του προγράμματος της άσκησης 1 που γράψατε? από πού προέκυψε αυτή η πληροφορία ?
7) Πόσα byte καταλαμβάνει στη μνήμη η μεταβλητή y ? και ποια είναι η τιμή του κάθε byte στη μνήμη (από τη μικρότερη προς τη μεγαλύτερη θέση μνήμης) ? γιατί αποθηκεύονται με αυτόν τον τρόπο ?
8) Για την μεταβλητή z, ποια είναι η τιμή του κάθε byte στη μνήμη (από τη μικρότερη προς τη μεγαλύτερη θέση μνήμης) ? προκύπτει η τιμή -17 και πως ? τι είδους μετατροπή συμβαίνει και που λαμβάνει χώρα αυτή ? (από τον προγραμματιστή ή από τον assembler ή από τη CPU) και γιατί ?