ΠΑΝΕΠΙΣΤΗΜΙΟ
ΔΥΤΙΚΗΣ ΜΑΚΕΔΟΝΙΑΣ

Αρχιτεκτονική Υπολογιστών

Ασκήσεις Εργαστηρίου

Ενότητα: ΕΡΓΑΣΤΗΡΙΑΚΗ ΑΣΚΗΣΗ No 11

Δρ. Μηνάς Δασυγένης

mdasyg@ieee.org

Τμήμα Μηχανικών Πληροφορικής και Τηλεπικοινωνιών

Εργαστήριο Ψηφιακών Συστημάτων και Αρχιτεκτονικής Υπολογιστών

http:// arch.icte.uowm.gr/mdasyg

Άδειες Χρήσης

- Το παρόν εκπαιδευτικό υλικό υπόκειται σε άδειες χρήσης Creative Commons.
- Για εκπαιδευτικό υλικό, όπως εικόνες, που υπόκειται σε άλλου τύπου άδειας χρήσης, η άδεια χρήσης αναφέρεται ρητώς.

Χρηματοδότηση

- Το παρόν εκπαιδευτικό υλικό έχει αναπτυχθεί στα πλαίσια του εκπαιδευτικού έργου του διδάσκοντα.
- Το έργο «**Ανοικτά Ψηφιακά Μαθήματα του Πανεπιστημίου Δυτικής Μακεδονίας**» έχει χρηματοδοτήσει μόνο τη αναδιαμόρφωση του εκπαιδευτικού υλικού.
- Το έργο υλοποιείται στο πλαίσιο του Επιχειρησιακού Προγράμματος «Εκπαίδευση και Δια Βίου Μάθηση» και συγχρηματοδοτείται από την Ευρωπαϊκή Ένωση (Ευρωπαϊκό Κοινωνικό Ταμείο) και από εθνικούς πόρους.

Περιεχόμενα

1. Σκοπός της άσκησης

- Πέρασμα παραμέτρων σε συναρτήσεις.
- Συνάρτηση απολύτου.
- Συνάρτηση μέγιστης τιμής.
- Συνάρτηση ταξινόμησης bubble-sort.
- Θέματα Ασφάλειας.

(A) 8 Ερωτήσεις
(C) 10 Ασκήσεις
(B) 1 Bonus

2. Εργαστηριακή άσκηση

(στην τελευταία σελίδα υπάρχουν βοηθητικές σημειώσεις)

1. Κατασκευάστε τη συνάρτηση εύρεσης απόλυτης τιμής της διαφοράς δύο αριθμών **|x1-x2|** με τους παρακάτω τρόπους (με τα συγκεκριμένα ονόματα):

- **(C1) abs_register:** πέρασμα παραμέτρων με καταχωρητές με τεχνική pass-by-value
- **(C2) abs_dataseg:** πέρασμα παραμέτρων με διευθύνσεις μνήμης δηλωμένες στο data segment με τεχνική pass-by-value
- **(C3) abs_stack:** πέρασμα παραμέτρων με σωρό και τεχνική pass-by-value
- **(C4) abs_dataseg_reference:** πέρασμα παραμέτρων με την τεχνική pass-by-reference. Οι διευθύνσεις μνήμης θα μπουν σε καταχωρητές.

Στο κυρίως πρόγραμμα θα κάνετε τα εξής για κάθε διαδικασία abs.

1. Θα τοποθετείτε κατάλληλα τις παραμέτρους.

2. Θα τοποθετείτε κατάλληλα σχόλια.

3. Θα κάνετε κλήση της συνάρτησης.

4. Θα παίρνετε το αποτέλεσμα που επιστρέφει η συνάρτηση *(δηλαδή την απόλυτη τιμή) και θα το εκτυπώνετε καλώντας την συνάρτηση εκτύπωσης διψήφιων δεκαδικών αριθμών (είχε αναπτυχθεί σε προηγούμενο εργαστήριο).*

(A1) Από τους κώδικες C1 έως C4, ποιος κώδικας έχει τη μεγαλύτερη χρήση σε σωρό;

(A2) Από τους κώδικες C1 έως C4, ποιος κώδικας έχει τη μεγαλύτερη χρήση σε καταχωρητές;

(A3) Από τους κώδικες C1 έως C4, ποιος κώδικας έχει τις περισσότερες εντολές assembly;

(A4) Από τους κώδικες C1 έως C4, ποιος κώδικας έχει τα περισσότερα byte σε machine code;

2. Κατασκευάστε τη συνάρτηση εύρεσης μέγιστης τιμής 6 αριθμών ως εξής:
    - **(C5) max_six_register:** πέρασμα παραμέτρων με καταχωρητές με τεχνική pass-by-value
    - **(C6) max_six_dataseg:** πέρασμα παραμέτρων με διευθύνσεις μνήμης δηλωμένες στο data segment με τεχνική pass-by-value
    - **(C7) max_six_stack:** πέρασμα παραμέτρων με σωρό και τεχνική pass-by-value
    - **(C8) max_six_dataseg_reference:** πέρασμα παραμέτρων με την τεχνική pass-by-reference. Η διεύθυνση μνήμης του πίνακα των 6 αριθμών θα μπει σε ένα καταχωρητή
    - **(C9) max_six_stack_reference:** πέρασμα παραμέτρων με την τεχνική pass-by-reference. Η διεύθυνση μνήμης του πίνακα των 6 αριθμών θα μπει στο σωρό.

(A5) Από τους κώδικες C5 έως C9, ποιος κώδικας έχει τη μεγαλύτερη χρήση σε σωρό;

(A6) Από τους κώδικες C5 έως C9, ποιος κώδικας έχει τη μεγαλύτερη χρήση σε καταχωρητές;

(A7) Από τους κώδικες C5 έως C9, ποιος κώδικας έχει τις περισσότερες εντολές assembly;

(A8) Από τους κώδικες C5 έως C9, ποιος κώδικας έχει τα περισσότερα byte σε machine code;

3. **(C10)** Κατασκευάστε τη συνάρτηση αύξουσας ταξινόμησης 10 αριθμών με τη τεχνική bubble sort με όποιον τρόπο θέλετε. Οι 10 αριθμοί είναι δηλωμένοι στο data segment ως πίνακας 10 θέσεων. Για παράδειγμα:
   ```
   array1 db 10,1,4,145,6,67,8,9,43,44
   ```

Η συνάρτηση αύξουσας ταξινόμησης **bubble sort** για n στοιχεία λειτουργεί ως εξής:

1. Ξεκινάμε από το στοιχείο στη θέση SI=0.
2. Το συγκρίνουμε με το στοιχείο στη θέση SI=SI+1, δηλαδή με το στοιχείο 1.
3. Αν το στοιχείο στη θέση SI=SI+1 είναι μεγαλύτερο τότε δε γίνεται τίποτα.
4. Αν το στοιχείο στη θέση SI=SI+1 είναι μικρότερο τότε γίνεται αμοιβαία αλλαγή των δυο στοιχείων. Δηλαδή, το στοιχείο στη θέση SI πηγαίνει στη θέση SI+1 και το στοιχείο στη θέση SI+1 πηγαίνει στη θέση SI.
5. Αυξάνεται το SI κατά 1 και ομοίως ελέγχουμε με το επόμενο.
6. Το SI αυξάνεται συνεχώς έως n-1 και επαναλαμβάνονται κάθε φορά τα βήματα 2 έως 5.
7. Μόλις εξεταστούν όλα τα στοιχεία, η διαδικασία επαναλαμβάνεται για n φορές.

Στο παραπάνω παράδειγμα ο πίνακας array1 θα έχει τιμές σε αύξουσα ταξινόμηση, ως εξής: 1,4,6,8,9....

3. Προχωρημένες Τεχνικές για κατανόηση του ασφαλούς προγραμματισμού

Ερωτήσεις Bonus

(B1 bonus +0,25)

Σας δίνεται το παρακάτω πρόγραμμα το οποίο ζητάει έναν κωδικό και στη συνέχεια επιβεβαιώνει κατά πόσο είναι σωστός ο κωδικός ή όχι, με βάση κάποιον κωδικό που έχει ήδη αποθηκευμένο στη μνήμη. Κάντε το compile και εκτελέστε το.

Κώδικας segment

```
KODIKAS SEGMENT
ARXH:
      MOV AX,DEDOMENA
      MOV DS,AX
      call printprompt
      call inputpass
      call checkpass

      MOV AH,4CH
      INT 21H

      printprompt   proc
            lea dx,prompt
            mov ah,09
            int 21h
            ret
      printprompt   endp

      loginok   proc
            pusha
            lea dx,loginokprompt
            mov ah,09
            int 21h
            popa
            ret
      loginok   endp

      loginfail   proc
            pusha
            lea dx,loginfailprompt
            mov ah,09
            int 21h
            popa
            ret
      loginfail   endp

      inputpass   proc
            mov si,0
            nextstr:
            mov ah,08
            int 21h
```

Κώδικας segment

```
            mov buffer[si],al
            inc si
            cmp al,13
            je _end_of_nextstr
            jmp nextstr
            _end_of_nextstr:
            ret
      inputpass endp

      checkpass proc
            pusha
            mov okpass,0
            mov si,0
            mov cx,7
            _check_pass:
            mov al,buffer[si]
            cmp al,password[si]
            je nextloop
            mov  okpass,1

            nextloop:
            inc si
            loop _check_pass:
            cmp okpass,0
            jne checkpass_failpass
            call loginok
            jmp _checkpassend
            checkpass_failpass:
            call loginfail
            _checkpassend:
            popa
            ret
      checkpass endp
      db 1000 dup(0)
      rooted proc
            pusha
            lea dx,rootedprompt
            mov ah,09
            int 21h
            popa
            ret
      rooted endp
KODIKAS ENDS
DEDOMENA SEGMENT
      db 186,008,000,180,009,205,033
      okpass db 0
      rootedprompt db  10,13,"Computer Rooted$",10,13
      loginokprompt db  10,13,"Correct Password$",10,13
      loginfailprompt db 10,13,"Bad Password$",10,13
      prompt DB 10,13,'Password:','$'
      buffer db 10 dup(0)
      password db "password"
DEDOMENA ENDS
SOROS SEGMENT STACK
```

Κώδικας segment

```
      DB 10 DUP(2)
SOROS ENDS
END ARXH
```

1. Το παραπάνω πρόγραμμα αν και λειτουργεί, εντούτοις έχει ένα λάθος προγραμματισμού στην ασφάλεια, που μπορεί να χρησιμοποιηθεί από έναν κακόβουλο χρήστη. Ποιο είναι αυτό το σφάλμα και πως είναι η επίσημη ονομασία *(ελληνικά και αγγλικά);*
2. Εκμεταλλευτείτε το συγκεκριμένο σφάλμα ασφάλειας καθώς εκτελείτε το πρόγραμμα ***(χωρίς να τροποποιήσετε τον κώδικα)***. Θα πρέπει να δώσετε διαφορετικό κωδικό από το **password** *(δηλαδή, χωρίς να γνωρίζετε τον κωδικό)* και εντούτοις να σας εκτυπωθεί το μήνυμα "`Correct Password`"**.** Περιγράψτε τι κάνατε.
3. Το σφάλμα που έχει το πρόγραμμα, μπορεί με κατάλληλη είσοδο, να συμπεριφερθεί απροσδόκητα, δηλαδή ύστερα από την κλήση μιας συνάρτησης, να μην επιστρέψει στη διεύθυνση που θα έπρεπε, αλλά σε μια άλλη διεύθυνση που δε θα έχει κανένα κώδικα, με συνέπεια να εκτελείται συνεχώς μια άσχετη εντολή. Περιγράψτε πως γίνεται αυτό και γιατί.
4. Αμέσως πριν το RET της συνάρτησης `printprompt`, προσθέστε τις παρακάτω γραμμές, όπου _____ είναι ένας αριθμός.

   MOV AX,____

   PUSH AX

   Με τις παραπάνω γραμμές θα πρέπει να κάνετε "`πειρατεία του IP`" *(IP hijack)* και αντί να επιστρέψει στη διεύθυνση που έπρεπε να το οδηγήσετε στην εκτέλεση της συνάρτησης rooted. Περιγράψτε τι κάνετε και γιατί *(**tip**: βρείτε το IP που αντιστοιχεί στη συνάρτηση rooted, δηλαδή τη διεύθυνση που ξεκινάει η συνάρτηση rooted)*
5. Μέσα στο Data Segment στη διεύθυνση 0 έχει γραφτεί ένα παράξενο αλφαριθμητικό. Όπως μπορεί να φανταστεί κάποιος, έχουν γραφεί bytes που αντιστοιχούν σε machine code *(αυτό ονομάζεται payload από τους κακόβουλους χρήστες).* Προσπαθήστε να γράψετε από 1 έως 5 εντολές assembly στο κυρίως πρόγραμμα, αμέσως μετά την αρχικοποίηση του καταχωρητή DS και πριν από την κλήση της πρώτης συνάρτησης, οι οποίες θα έχουν ως συνέπεια να μπερδευτεί ο επεξεργαστής και να εκτελέσει τα bytes που βρίσκονται στο data segment, ως machine code εντολές assembly. Αμέσως μετά την εκτέλεση του payload το πρόγραμμα θα κολλήσει *(γιατί);*
6. Τοποθετήστε ένα δικό σας payload *(δηλαδή, μια σειρά από bytes machine code είτε στο δεκαδικό είτε στο δεκαεξαδικό σύστημα)* στη θέση αυτού του payload, που θα έχει ως συνέπεια να γίνεται διαίρεση με το 0, ώστε να δημιουργηθεί η εξαίρεση **DIVISION BY ZERO.**
7. Προσπαθήστε να τοποθετήστε συγκεκριμένους χαρακτήρες κατά την εισαγωγή, ώστε να υπερχειλίσει η περιοχή δεδομένων και να συνεχίσουν να γράφονται οι χαρακτήρες πάνω στο σωρό. Στη συνέχεια αφού πατήσετε enter, θα εκτελεστεί το RET και θα έχει ως συνέπεια να διαβαστεί από το

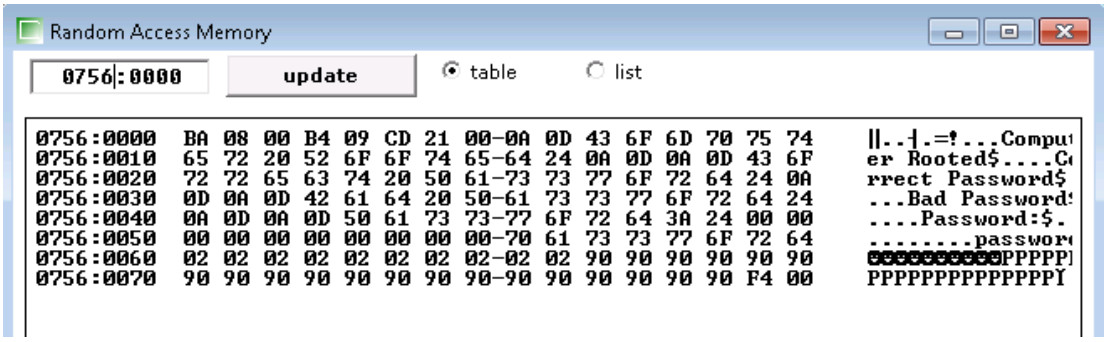
σωρό η διεύθυνση επιστροφής. Αυτή όμως θα έχει διαγραφεί, αφού θα έχουν τοποθετηθεί τα bytes της εισόδου.
➔ Δώστε κατάλληλη είσοδο, ώστε η επιστροφή να οδηγηθεί στη συνάρτηση `routed`.

- **tip1:** βρείτε πρώτα το IP που αντιστοιχεί στη συνάρτηση routed
- **tip2:** κάθε χαρακτήρας που βάζετε στην είσοδο είναι 1 byte, όποτε όταν γεμίσει το τμήμα δεδομένων και συνεχίσει προς το σωρό, θα πρέπει να βρείτε τα 2 byte *(δηλαδή τους 2 χαρακτήρες)* που θα δημιουργήσουν τη διεύθυνση επιστροφής που θα γραφτεί στο σωρό.
- **tip3:** χρησιμοποιήστε τις διευθύνσεις έναρξης του τμήματος δεδομένων (**DS**) και του τμήματος σωρού (**SS**), προκειμένου να σας βοηθήσει να υπολογίσετε τους χαρακτήρες που πρέπει να τοποθετήσετε.

Για να βοηθηθείτε, χρησιμοποιήστε το παράθυρο της μνήμης από το emulator με **View → Memory.**

Για παράδειγμα, αν στο DS δείτε την τιμή **0756**, τότε πηγαίνετε στο παράθυρο αυτό και βάζετε την τιμή `0756:0000` και πατάτε UPDATE. Βλέπετε τα περιεχόμενα της μνήμης από αυτή τη διεύθυνση και παραπάνω. Θα δείτε τα μηνύματα του Data Segment, και στη συνέχεια τα περιεχόμενα του σωρού.

Στην παρακάτω εικόνα μπορούμε εύκολα να διακρίνουμε που ξεκινάει ο σωρός, αφού κατά τη δήλωση του σωρού είχε δοθεί 10 dup(2) δηλαδή αρχική τιμή 2 που αντιστοιχεί στον ASCII χαρακτήρα με το μαύρο πρόσωπο ☻



4. Ερωτήσεις & Απαντήσεις για καλύτερη κατανόηση

1. Ποιοι καταχωρητές χρησιμοποιούνται στην τεχνική pass-by-value;

Μπορεί να χρησιμοποιηθεί οποιοσδήποτε καταχωρητής είτε 16bit είτε 8bit.
Μπορούμε να χρησιμοποιήσουμε δηλαδή τους:
AX, BX, CX, DX, SI, DI, BP, AH, AL, BH, BL, CH, CL, DH, DL.

Αν θέλουμε να μεταφέρουμε δεδομένα που χωράνε σε 8bit, μπορούμε να χρησιμοποιήσουμε τους 8bit καταχωρητές, αρκεί να χρησιμοποιήσουμε σωστά το push *(push γίνεται μόνο σε 16bit).*

ΠΑΡΑΔΕΙΓΜΑ:
mov AL,number1 ;πρώτη παράμετρος
mov AH,number2 ;δεύτερη παράμετρος
call my_procedure

2. Πως βρίσκεται η απόλυτη τιμή της διαφοράς δυο αριθμών;

Ψευδοκώδικας:
Έστω x1,x2 οι αριθμοί.
Αν x1>x2 τότε η απόλυτη τιμή ισούται με x1-x2
Αν x1<x2 τότε η απόλυτη τιμή ισούται με x2-x1

3. Που βρίσκονται οι αριθμοί στη συνάρτηση max_six_??

Οι αριθμοί είναι δηλωμένοι στο data segment ως εξής:

```
numbers 6,5,10,1,20
```

μπορούμε να χρησιμοποιήσουμε
τους αριθμούς ως

numbers[0], numbers[1] κ.ο.κ.

4. Πως γίνονται οι συγκρίσεις στη συνάρτηση αυτή;

(α) τοποθετούμε τα στοιχεία σε 6 καταχωρητές.

(β) τοποθετούμε τις διευθύνσεις τον στοιχείων σε 6 καταχωρητές

(γ-δ-ε) γνωρίζουμε μόνο τη διεύθυνση της αρχής του πίνακα και τον αριθμό των στοιχείων.

Χρησιμοποιούμε τον παρακάτω **ψευδοκώδικα**:

max=numbers[0]

Από i=1 έως 6

An max>number[i] τότε επόμενο i

An max \< number[i] τότε max=number[i]