Αρχιτεκτονική Υπολογιστών

Ενότητα 6: Διαδικασίες, Σωρός, Διαφανείς συναρτήσεις

Δρ. Μηνάς Δασυγένης
mdasyg@ieee.org
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 στο Πανεπιστήμιο 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.

Υποπρογράμματα (1/3)

- Τα υποπρογράμματα είναι τμήματα κώδικα που συγκεντρώνουν ένα σύνολο από εντολές το οποίο θα εκτελέσει πλήρως μια λειτουργία και πιθανόν να επιστρέψει και τιμή.
- Τα υποπρογράμματα είναι απαραίτητα στοιχεία του δομημένου προγραμματισμού Δεδομένου ότι κάθε υποπρόγραμμα μπορεί να γραφτεί, να ελεγχθεί και να δοκιμαστεί ανεξάρτητα από τα υπόλοιπα, είναι ευκολότερη η υλοποίηση και η επαλήθευση ενός προγράμματος που αποτελείται από υποπρογράμματα.
- Η διάσπαση ενός προγράμματος σε υποπρογράμματα κάνει το πρόγραμμα εύκολο κατανοητό γιατί το κάθε υποπρογράμματα μπορεί να διαβαστεί και να κατανοηθεί ξεχωριστά.

Υποπρογράμματα (2/3)

- Ένα υποπρόγραμμα μπορεί να κληθεί από διάφορα σημεία του προγράμματος, με διαφορετικές αν χρειάζεται παραμέτρους.
- Μπορεί επίσης για λόγους αυτονομίας να χρησιμοποιεί τοπικές μεταβλητές οι οποίες υπάρχουν μόνο όσο χρόνο εκτελείται το υποπρόγραμμα αυτό.
- Το πρόγραμμα αποτελείται από το κύριο τμήμα το οποίο καλεί διάφορα υποπρογράμματα τα όποια με την σειρά τους πιθανόν να καλούν και αλλά υποπρογράμματα.
- Κάθε υποπρόγραμμα έχει συγκεκριμένο όνομα, η αρχή του δηλώνεται με την οδηγία PROC και το πέρας του με την οδηγία ENDP. Το όνομα του υποπρογράμματος είναι ετικέτα που αντιπροσωπεύει την λογική δ/νση μνήμης της αρχής του κώδικα του υποπρογράμματος.

Υποπρογράμματα (3/3)

- Κάθε υποπρόγραμμα καλείται με την εντολή CALL και το όνομά του Π.χ.

```
CALL  emfanish__char
```

- Κάθε υποπρόγραμμα όταν ολοκληρώσει τη λειτουργία του επιστρέφει στο κυρίως πρόγραμμα ή στο υποπρόγραμμα που το κάλεσε με την εντολή RET.
- Και η επιστροφή στο λειτουργικό σύστημα γίνεται με την εντολή RET.
- Η ύπαρξη σωρού είναι απαραίτητη για την λειτουργία των υποπρογραμμάτων.
- Ο σωρός είναι μια περιοχή μνήμης ειδικά δεσμευμένη για τη λειτουργία σωρού.
- Η δέσμευση της μνήμης του σωρού στην asssembly γίνεται με ειδικές εντολές ή μπορεί να πραγματοποιηθεί αυτόματα από κάποιους assembler. Για λόγους συμβατότητας με κάθε assember, εμείς πάντα θα ορίζουμε μνήμη σωρού.

Υλοποίηση διαδικασιών στη x86 (1)

- Απαιτούνται:
  - Να οριστεί ένα τμήμα μνήμης για το σωρό για την αποθήκευση της δ/νσης επιστροφής της συνάρτησης:

```
SOROS SEGMENT STACK
   db 256 dup(0)
 SOROS ENDS
```

- Ο κώδικας της συνάρτησης ξεκινάει με **(όνομα συνάρτησης) PROC** και τελειώνει με **RET** (επιστροφή) και **(όνομα συνάρτησης) ENDP.**

```
SYNARTHSHMOU PROC
Κώδικας
RET
SYNARTHSHMOU ENDP
```

ΕΝΤΟΛΕΣ **CALL** ΚΑΙ **RET**

Η λειτουργία των υποπρογραμμάτων επιτυγχάνεται με το συνδυασμό των εντολών CALL και RET και με την βοήθεια του σωρού, κατά τον εξής τρόπο:

- Η εντολή CALL αποθηκεύει αυτόματα την δ/νση της εντολής μετά την CALL *(**της επόμενης εντολής**)* στο σωρό, και μεταφέρει με άλμα τον έλεγχο του προγράμματος στην δ/νση μνήμης του υποπρογράμματος της CALL *(αυτό επιτυγχάνεται με το να γράψει στον ειδικό καταχωρητή εντολών IP τη διεύθυνση μνήμης της πρώτης εντολής του υποπρογράμματος).*
- Μόλις το υποπρόγραμμα ολοκληρώσει τη λειτουργία εκτελεί την εντολή RET, η οποία ανακαλεί από τον σωρό την δ/νση επιστροφής και την φορτώνει στον μετρητή προγράμματος (IP) για να συνεχιστεί η εκτέλεση του προγράμματος στην εντολή μετά την CALL που κάλεσε το υποπρόγραμμα.

FAR ή NEAR (1/2)

- Η αρχιτεκτονική x86 χωρίζει τη μνήμη σε τμήματα (~ομάδες διευθύνσεων).
- Οι εντολές ή τα δεδομένα ενός προγράμματος μπορεί να βρίσκονται στο ίδιο τμήμα (ενδοτμηματικά) ή σε άλλο τμήμα (εξωτμηματικά).
- Ως προς τις εντολές, αυτές θεωρούνται ότι βρίσκονται στο ίδιο τμήμα, αν έχουν ίδια τιμή στον ειδικό καταχωρητή τμήματος κώδικα CS. Αυτές οι εντολές έχουν ίδια τιμή στο CS, αλλά διαφορετική τιμή στο IP.
- Οι εντολές που βρίσκονται σε διαφορετικά τμήματα, έχουν διαφορετική τιμή στο CS. Η τιμή της μετατόπισης (τιμή του IP) κάποιας εντολής μπορεί να είναι ίδια ή διαφορετική.

FAR ή NEAR (2/2)

- Ως προς τα δεδομένα, αυτά θεωρούνται ότι βρίσκονται στο ίδιο τμήμα, αν έχουν ίδια τιμή στον ειδικό καταχωρητή τμήματος κώδικα DS.
- Η μετατόπιση των δεδομένων μέσα στο τμήμα κώδικα, υπολογίζεται από τους ειδικούς καταχωρητές SI,DI.
- Κάθε υποπρόγραμμα είναι NEAR ή FAR.
  - NEAR σημαίνει ότι ο κώδικας του βρίσκεται στο ίδιο τμήμα με τον κώδικα που χρησιμοποιεί το υποπρόγραμμα (δηλαδή, ο κώδικας που εκτελεί τη κλήση CALL) και έτσι και τα 2 κομμάτια κώδικα έχουν την ίδια τιμή στο CS.
  - FAR σημαίνει ότι η περιοχή μνήμης που περιέχει το υποπρόγραμμα, βρίσκεται σε διαφορετικό τμήμα και άρα έχει διαφορετική τιμή στο CS, από τον καλών κώδικα.

**** **Ο assembler καταλαβαίνει και τοποθετεί τις αντίστοιχες εντολές για FAR ή NEAR.** ****

Υλοποίηση διαδικασιών στη x86 (2)

- Στο σημείο που θέλουμε να καλέσουμε τη συνάρτηση, την καλούμε με την εντολή CALL *(όνομα συνάρτησης).*

```
κώδικας
CALL SYNARTHSHMOU
κώδικας
```

- Αν η συνάρτηση τροποποιεί καταχωρητές επιβάλλεται να αποθηκεύσουμε τις τρέχουσες τιμές που έχουν στην είσοδο της συνάρτησης και να τις επαναφέρουμε στην έξοδο. Μπορούμε να τις αποθηκεύσουμε σε θέσεις μνήμης ή στο σωρό.

ΣΩΡΟΣ (STACK)

- **Καταχωρητής τμήματος σωρού SS (Stack Segment)**
  Ο καταχωρητής SS περιέχει την δ/νση από όπου αρχίζει το τμήμα μνήμης της σωρού (STACK).

- **Δείκτης βάσης BP (Base Pointer)**
  Χρησιμοποιείται για προσπέλαση δεδομένων στο σωρό (τοπικές μεταβλητές, παράμετροι υποπρογραμμάτων).

- **Δείκτης σωρού SP (Stack Pointer)**
  Δείχνει την πρώτη ελεύθερη θέση στο σωρό. Δηλαδή το σημείο μέχρι το οποίο έχει γεμίσει ο σωρός

  Π.χ. αν SS = 1ABCh και SP = 100h, τότε η φυσική δ/νση της κορυφής του σωρού είναι:

  SS * 10h + SP = 1ABCh * 10h + 100h = 1ABC0h + 100h = 1ACC0h

Εντολές CALL (1/4)

- Όπως αναφέρθηκε υπάρχουν δυο τύποι κλήσεων με την εντολή CALL, η ενδοτμηματική και η εξωτμηματική. Ομοίως, υπάρχουν δυο τύποι της εντολής επιστροφής RET.
- Ο προγραμματιστής χρησιμοποιεί μόνο τις εντολές CALL/RET και προαιρετικά μόνο προσθέτει στην ίδια γραμμή τις λέξεις FAR/NEAR, κάτι που δε συνιστάται.
- Όταν το υποπρόγραμμα που βρίσκεται η CALL είναι NEAR πρόκειται για ενδοτμηματική CALL (near ptr call), αλλιώς αν είναι FAR τότε η CALL είναι εξωτμηματική (far ptr call) σε άλλο code segment.
- Στην ενδοτμηματική κλήση μεταφέρεται στην κορυφή του σωρού το περιεχόμενο του IP και μειώνεται ο δείκτης σωρού SP κατά 2. Ακολούθως συνεχίζεται η εκτέλεση του προγράμματος στην δ/νση του υποπρογράμματος, η οποία όμως πρέπει να βρίσκεται στο ίδιο τμήμα κώδικα.

Εντολές CALL (2/4)

- Στην εξωτμηματική κλήση μεταφέρεται πρώτα στην κορυφή του σωρού το περιεχόμενο του καταχωρητή τμήματος κώδικα CS και μειώνεται ο δείκτης σωρού SP κατά 2 και ακολούθως μεταφέρεται στην κορυφή του σωρού το περιεχόμενο του IP και μειώνεται πάλι ο δείκτης σωρού SP.
- Η εκτέλεση του προγράμματος συνεχίζεται στην δ/νση του υποπρογράμματος, η οποία μπορεί να βρίσκεται σε διαφορετικό τμήμα κώδικα από το τμήμα εντολής CALL.
- Στην περίπτωση αυτή πριν από το όνομά του υποπρογράμματος προαιρετικά τοποθετείται η έκφραση FAR PTR αν δεν υπάρχουν οι δηλώσεις NEAR ή FAR
  π.χ. **`CALL FAR PTR emfanish__char`**
- Στους σύγχρονους compiler αυτό γίνεται αυτόματα και δεν απαιτείται από το χρήστη.

Εντολές CALL (3/4)

Υπάρχουν δυο τύποι της εντολής επιστροφής RET.

- Όταν το υποπρόγραμμα που βρίσκεται η RET είναι NEAR πρόκειται για ενδοτμηματική RET (RETN), αλλιώς αν είναι FAR τότε η RET είναι εξωτμηματική (RETF).
- Στην ενδοτμηματική RET μεταφέρεται από την κορυφή του σωρού το περιεχόμενο στον IP και αυξάνεται ο δείκτης σωρού SP κατά 2. Ακολούθως συνεχίζεται η εκτέλεση του προγράμματος στη μετά την CALL εντολή.

Εντολές CALL (4/4)

- Στην εξωτμηματική RET μεταφέρεται από την κορυφή του σωρού το περιεχόμενο του IP αυξάνεται ο δείκτης σωρού SP κατά 2 και ακολούθως μεταφέρεται από νέα κορυφή σωρού το περιεχόμενο στον CS αυξάνεται πάλι ο δείκτης σωρού SP κατά 2 και συνεχίζεται η εκτέλεση του προγράμματος.

- Όπως φαίνεται από τα παραπάνω πρέπει ο συνδυασμός των εντολών CALL και RET να γίνεται με προσοχή, ούτως ώστε και οι δυο να είναι του ίδιου τύπου δηλαδή ή ενδοτμηματικές ή εξωτμηματικές.

FAR/NEAR CALL/RET (1/2)

- Ο προγραμματιστής τοποθετείται μόνο τις λέξεις CALL ή RET και ο compiler αντιλαμβάνεται αν είναι FAR ή NEAR.
- Η οδηγία NEAR έχει εξαλειφθεί και πια δεν υποστηρίζεται από το assembler (θα υπάρχει συντακτικό λάθος στο call near ptr myprocedure).
- Η διαφορά του FAR ή NEAR φαίνεται και με το διαφορετικό MACHINE CODE για την εντολή CALL.

```
25]   012E: E8 0C 00          call myprocedure
26]   0131: FF 00 1D          call far ptr myprocedure
27]      :
```

FAR/NEAR CALL/RET (2/2)

```
TITLE ASKISI
KODIKAS SEGMENT PUBLIC
ASSUME CS:KODIKAS, DS:DEDOMENA, SS:SOROS
MAIN PROC NEAR
    MOV AX,DEDOMENA          ; Apokatastash tou DS
    MOV DS,AX                ; Apokatastash tou DS
    ..........
    CALL DISPLAY-HEX         ; Kaloume tin Display_hex
    MOV AH,4CH               ; Eksodos sto leitoyrgiko systhma
    INT 21H
MAIN ENDP
DISPLAY-HEX PROC NEAR
    ..........
    CALL ONE-DIGIT           ; Kaloume tin One_digit
    ..........
    RET                      ; Epistrefo stin thesi apo opou klithike h yporoutina
DISPLAY-HEX ENDP
ONE-DIGIT PROC NEAR
    ..........
TELOS:   RET                 ; Epistrefo stin thesi apo opou klithike h yporoutina
ONE-DIGIT ENDP
KODIKAS ENDS
DEDOMENA SEGMENT
    ......
DEDOMENA ENDS
SOROS SEGMENT STACK
    DB 256 DUP(0)
SOROS ENDS
    END MAIN
```

Πέρασμα παραμέτρων

- Η ενέργεια με την οποία μεταβιβάζουμε δεδομένα διαφορετικά κάθε φορά στα υποπρογράμματα για να τα καθοδηγήσουμε τι να κάνουν, λέγεται πέρασμα παραμέτρων.
- Υπάρχουν δύο μέθοδοι για πέρασμα παραμέτρων:
  - Πέρασμα μέσω καταχωρητών.
  - Πέρασμα μέσω σωρού.
- Η μέθοδος του περάσματος μέσω καταχωρητών είναι πιο απλή και χρησιμοποιείται σε προγράμματα αποκλειστικά σε γλώσσα ASSEMBLY.
- Η μέθοδος του περάσματος μέσω σωρού είναι σχετικά πιο δυσχερής μέθοδος, απαιτεί περισσότερο κώδικα άλλα είναι πολύ ευέλικτη αφού μπορούμε να περάσουμε πάρα πολλούς παραμέτρους χωρίς τον περιορισμό του αριθμού των καταχωρητών. Για αυτό το λόγο την χρησιμοποιούν όλοι οι σύγχρονοι compilers.

Πέρασμα μέσω καταχωρητών

- Στην περίπτωση αυτή επιλέγονται οι κατάλληλοι καταχωρητές σαν παράμετροι οι οποίοι βέβαια είναι οι ίδιοι σε κάθε κλήση του υποπρογράμματος.
- Γι' αυτόν τον λόγο πρέπει πριν από τον κώδικα κλήσης του υποπρογράμματος, με κατάλληλα σχόλια, να αναγράφονται όλοι οι καταχωρητές που χρησιμοποιούνται σαν παράμετροι.

Π.χ.

```
MOV DL,AL ;O DL GIA PERASMA sto proc
CALL EMFANISH_CHAR
```

Πέρασμα μέσω σωρού

- Στη περίπτωση αυτής της μεθόδου περάσματος, το πρόγραμμα που καλεί μεταφέρει στον σωρό με εντολές PUSH τις παραμέτρους, ενώ το υποπρόγραμμα ανακτά από το σωρό τις παραμέτρους με τη βοήθεια του καταχωρητή δείκτη BP.
- Μετά την επιστροφή από το υποπρόγραμμα (μετά την εντολή CALL) πρέπει να αποκαθίσταται ο σωρός.
- Η χρήση της μεθόδου είναι υποχρεωτική εφόσον το υποπρόγραμμα πρόκειται να συνδεθεί και να κληθεί από πρόγραμμα υψηλότερων επιπέδων (π.χ. C).

Π.χ.

```
LEA AX,MINIMA
PUSH AX ;O AX MPAINEI STI SORO GIA PERASMA
CALL EMFANISI_MINIMA
```

Διαφανείς Συναρτήσεις (1/3)

- Μια συνάρτηση η οποία τροποποιεί τους καταχωρητές και δεν τους επαναφέρει κατά την έξοδο ονομάζεται **ΜΗ ΔΙΑΦΑΝΗΣ ή ΑΔΙΑΦΑΝΗΣ** συνάρτηση.
- Αυτού του είδους οι συναρτήσεις προκαλούν πρόβλημα, ιδιαίτερα αν κληθούν από γλώσσες υψηλού επιπέδου.
- Θα πρέπει να αποθηκεύονται οι τιμές των καταχωρητών, είτε σε θέσεις μνήμης *(δεν είναι καλή λύση, γιατί αν κληθεί πάλι--αναδρομικά--η συνάρτηση θα έχουμε πρόβλημα)*, είτε στο σωρό *(προτιμότερο)*

Διαφανείς Συναρτήσεις (2/3)

Παράδειγμα αδιαφανούς συνάρτησης:

- Έστω η συνάρτηση:

```
PRINTMSG PROC
        LEA DX, MHNYMA
        MOV AH,09
        INT 21h
        RET
PRINTMSG ENDP
```

- Η συνάρτηση αυτή τροποποιεί τον καταχωρητή DX και στο τέλος δεν τον επαναφέρει στην τιμή που είχε πριν κληθεί.

*** Επίσης τροποποιείται και ο AX. ***

- Θα πρέπει λοιπόν να αποθηκευτεί προσωρινά η τιμή του DX στο σωρό και να την επαναφέρουμε πριν το RET.

Διαφανείς Συναρτήσεις (3/3)

Παράδειγμα Διαφανούς συνάρτησης:

Τοποθετούμε εντολές PUSH *(αμέσως μετά τη δήλωση της PROC)* και **POP** *(πριν το RET)* για κάθε καταχωρητή που τροποποιούμε:

```
PRINTMSG PROC
        PUSH DX
        PUSH AX
        LEA DX, MHNYMA
        MOV AH,09
        INT 21h
        POP AX
        POP DX
        RET
PRINTMSG ENDP
```

- Η σειρά των POP είναι αντίστροφη της σειράς των PUSH γιατί ο σωρός είναι μια δομή Last In First Out.
- PUSH, POP γίνονται μόνο σε καταχωρητές 16Bit.
- Για κάθε καταχωρητή απαιτείται PUSH, POP.

Template (B)
(Κοινός σκελετός αρχείου ASM)

```
TITLE MYPROGRAM
DEDOMENA SEGMENT
[.....εδώ πέρα τοποθετούνται τα
δεδομένα.....]
DEDOMENA ENDS

KODIKAS SEGMENT
ARXH: MOV AX,DEDOMENA
      MOV DS,AX
[....εδώ τοποθετείται ο κύριος
κώδικάς του προγράμματος...]
      MOV AH,4CH
      INT 21H
[....εδώ τοποθετούνται οι
συναρτήσεις...]
KODIKAS ENDS
SOROS SEGMENT STACK
        db 256 dup(0)
SOROS ENDS
END ARXH
```

- Όλα τα προγράμματα σε ASM 8086 με συναρτήσεις χρησιμοποιούν το διπλανό κώδικα ως σκελετό. Προσθέτετε δεδομένα και εντολές στις αντίστοιχες περιοχές.
- Υπάρχει σωρός για τις συναρτήσεις

Τέλος Ενότητας

Με τη συγχρηματοδότηση της Ελλάδας και της Ευρωπαϊκής Ένωσης