Αρχιτεκτονική Υπολογιστών

Ενότητα 13: Λειτουργίες Αρχείων

Δρ. Μηνάς Δασυγένης
mdasyg@ieee.org
Εργαστήριο Ψηφιακών Συστημάτων και Αρχιτεκτονικής Υπολογιστών
http://arch.ece.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.

Εισαγωγή στα αρχεία

- Ένα αρχείο δεν είναι τίποτα περισσότερο από μια σειρά από bytes αποθηκευμένα σε μια μονάδα περιφερειακής μνήμης (σκληρό δίσκο, δισκέτα, CD, κτλ).
- Στην Assembly η διαχείριση των αρχείων γίνεται με τη χρήση των interrupts (INT).
- Ο emulator υποστηρίζει μια σειρά από software interrupts που είναι συμβατά με όλους τους επεξεργαστές της σειράς x86 της Intel. Τα interrupts αυτά καλούνται με την εντολή INT και τον αριθμό του interrupt ενώ συνήθως την εντολή INT και τον αριθμό του interrupt ενώ συνήθως του interrupt.

Χειρισμός αρχείων

- ***MS-DOS INTERRUPTS (INT 21H) ΣΤΟΝ EMULATOR***

Τα interrupts του MS-DOS εκτελούνται από τον emulator σε προσομοίωση. Το σύστημα αρχείων του DOS προσομοιώνεται στον κατάλογο C:\emu8086\vdrive\x (x είναι το γράμμα μιας μονάδας δίσκου). Αν δεν καθορίζεται γράμμα μονάδας δίσκου και τρέχον κατάλογος τότε χρησιμοποιείται εξ ορισμού η διαδρομή:

C:\emu8086\MyBuild\.

Χειρισμός αρχείων στο vpack emu8086

- ***MS-DOS INTERRUPTS (INT 21H) ΣΤΟΝ EMULATOR***

Προσοχή: Αν χρησιμοποιείτε το emu8086.virtualpackage.exe τότε η διαδρομή των hardware interrupts δεν είναι η c:\emu8086.hw, αλλά η διαδρομή που αντιστοιχεί στο εικονικό περιβάλλον (φάκελος VOS, υποκατάλογος της εφαρμογής emu8086). Ο φάκελος VOS (virtual operating system) μπορεί να βρεθεί αν ανοίξετε ένα τερματικό windows (έναρξη->εκτέλεση->cmd) και στο μαύρο παράθυρο δώστε:

cd c:\

dir VOS /s /a

με τις παραπάνω εντολές θα σας εμφανιστεί που βρίσκεται ο φάκελος

VOS.

Χειρισμός αρχείων (vpack)

- ***MS-DOS INTERRUPTS (INT 21H) ΣΤΟΝ EMULATOR***

π.χ. Directory of C:\Users\user\AppData\Roaming. Μέσα στο φάκελο VOS θα βρείτε το φάκελο για την εφαρμογή (π.χ. emu8086 microprocessor emulator), και μέσα σε αυτό το φάκελο, θα βρείτε το φάκελο C_\ ο οποίος είναι ο εικονικός τοπικός δίσκος για τη εφαρμογή μας.

Οπότε, για παράδειγμά, η πλήρης διεύθυνση για τα interrupt είναι η: **C:\Users\user\AppData\Roaming\VOS\emu8086 microprocessor emulator\C_**

Χειρισμός αρχείων

- ***MS-DOS INTERRUPTS (INT 21H) ΣΤΟΝ EMULATOR***

Σημείωση: Το DOS περιορίζει το μέγεθος του ονόματος αρχείων και καταλόγων στους 8 χαρακτήρες και 3 χαρακτήρες επέκταση. Παράδειγμα έγκυρου ονόματος αρχείου είναι το: myfile.txt (όνομα αρχείου = 6 χαρακτήρες, επέκταση = 3 χαρακτήρες). Η επέκταση γράφεται μετά την τελεία και δεν επιτρέπονται άλλες τελείες.

Σκελετός αρχείου

- Στα εργαστήρια χρησιμοποιούμε split ή segment memory model (ξεχωριστά τμήματα για κάθε τύπο). Πιο συγκεκριμένα χρησιμοποιούμε DEDOMENA SEGMENT και KODIKAS SEGMENT για να διαχωρίσουμε τα δεδομένα από το κυρίως πρόγραμμα, ενώ ο σκελετός που δίνεται παρακάτω είναι τύπου flat memory model (δεδομένα και εντολές στο ίδιο τμήμα).

- Και οι δυο τρόποι μπορούν να χρησιμοποιηθούν για τη διαχείριση των αρχείων.

Σκελετός αρχείου

flat memory model

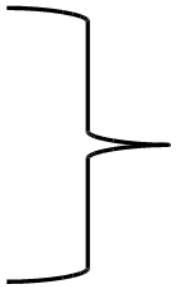
```
name "file“

jmp start

dir1 db "c:\test1", 0
file1 db "c:\test1\file1.txt", 0
handle dw ?
text db “χαρακτήρ.  που θα γραφτούν μέσα στο txt"
text_size = $ - offset text

start:
mov ax, cs
mov dx, ax
mov es, ax

[....εδώ τοποθετείται ο κύριος
κώδικάς των εντολών...]
```



[.....εδώ πέρα τοποθετούνται τα δεδομένα.....]

Δημιουργία Καταλόγου *(διεύθυνση- διαδρομή):*

- **INT 21h / AH= 39h**

Είσοδος: DS:DX -> String τερματιζόμενο με τον χαρακτήρα ASCII 0 string που δηλώνει το path.

- πχ.

```
; create c:\emu8086\vdrive\C\test1
mov dx, offset dir1
mov ah, 39h
int 21h
```

Τελεστής OFFSET και LEA

Σημείωση: Κατά την επεξεργασία τμημάτων μνήμης είναι απαραίτητος ο υπολογισμός της δ/νσης ετικέτας. Αυτό γίνεται με τον τελεστή OFFSET ή και με την εντολή:

LEA <προορισμός>, <ετικέτα>

Το ίδιο αποτέλεσμα μπορούμε να έχουμε με την χρήση του τελεστή OFFSET.

MOV <προορισμός>, OFFSET <ετικέτα>

- πχ.

  mov dx, offset dir1  LEA dx, dir1

Δημιουργία και άνοιγμα αρχείου: (1/)

- **INT 21h / AH= 3Ch**

Είσοδος: DS:DX -> Το όνομα του αρχείου σε ASCII.

CX = ιδιότητες αρχείου: mov cx, 0 ; normal - no attributes.

```
mov cx, 1 ; read-only.
mov cx, 2 ; hidden.
mov cx, 4 ; system
mov cx, 7 ; hidden, system and read-only!
mov cx, 16 ; archive
```

Επιστρέφει: CF=0 εάν επιτυχής, AX = file handle. CF=1 εάν όχι επιτυχής

AX = κωδικός σφάλματος.

Σημείωση: εάν το αρχείο υπάρχει διαγράφεται χωρίς προειδοποίηση!

Δημιουργία και άνοιγμα αρχείου: (2/)

- πχ.

```
; create and open file: c:\emu8086\vdrive\C\test1\file1.txt
mov ah, 3ch
mov cx, 0
mov dx, offset file1
int 21h
mov handle, ax
```

Άνοιγμα υπάρχοντος αρχείου:

- **INT 21h / AH= 3Dh**

Είσοδος: DS:DX -> Το όνομα του αρχείου σε ASCII.
AL = modes προσπέλασης και διαμοιρασμού:

```
 mov al, 0 ; read
 mov al, 1 ; write
 mov al, 2 ; read/write
```

Επιστρέφει CF=0 εάν επιτυχής, AX = file handle. CF=1 εάν όχι επιτυχής AX = κωδικός σφάλματος.

Σημείωση: ο δείκτης αρχείου πρέπει να είναι στην αρχή του αρχείου και το αρχείο να υπάρχει.

Εγγραφή δεδομένων σε αρχείο:

- **INT 21h / AH= 40h**

Είσοδος: BX = file handle. CX = αριθμός bytes προς εγγραφή. DS:DX -> δεδομένα προς εγγραφή.

Επιστρέφει CF=0 εάν επιτυχής - AX = αριθμός bytes που εγγράφηκαν πραγματικά. CF=1 εάν όχι επιτυχής AX = κωδικός σφάλματος.

Σημείωση: εάν CX=0, δεν γράφονται δεδομένα και το αρχείο μηδενίζεται. Η εγγραφή των δεδομένων γίνεται ξεκινώντας από την τρέχουσα θέση του αρχείου η οποία θέση ενημερώνεται μετά από μια επιτυχή εγγραφή. Συνήθης λόγος που AX < CX στην επιστροφή είναι γεμάτος δίσκος.

πχ.

```
; write to file:
mov ah, 40h
mov bx, handle
mov dx, offset text
mov cx, text_size
int 21h
```

Διάβασμα δεδομένων από αρχείο:

- **INT 21h / AH= 3Fh**

Είσοδος: BX = file handle. CX = αριθμός bytes που θα διαβαστούν.

DS:DX -> buffer για τα δεδομένα. Επιστρέφει: CF=0 εάν επιτυχής - AX

= αριθμός bytes που διαβάστηκαν πραγματικά; 0 εάν EOF (end of file) πριν την κλήση του interrupt. CF=1 εάν όχι επιτυχής AX = κωδικός σφάλματος.

Σημείωση: η ανάγνωση των δεδομένων ξεκινά από την τρέχουσα θέση του αρχείου και η θέση του αρχείου ενημερώνεται μετά από μια του αρχείου και η θέση του αρχείου ενημερώνεται μετά από μια την αρχή του αρχείου τότε η τιμή του AX θα είναι μικρότερη της τιμής του CX.

πχ.

```
; read from file
mov ah,3fh
mov bx, handle
mov dx, offset buffer
mov cx,01
int  21h
```

Κλείσιμο αρχείου:

- **INT 21h / AH= 3Eh**

Είσοδος: BX = file handle

Επιστρέφει: CF=0 εάν επιτυχής, CF=1 εάν όχι επιτυχής AX = κωδικός σφάλματος. (06h)

Πχ.

```
; close c:\emu8086\vdrive\C\test1\file1.txt
mov ah, 3eh
mov bx, handle
int 21h
```

Διαγραφή αρχείου:

- **INT 21h / AH= 41h**

Είσοδος: DS:DX -> Το όνομα του αρχείου σε ASCII (όχι χαρακτήρες μπαλαντέρ ?,!,*).

Επιστρέφει: CF=0 εάν επιτυχής, AL το drive του διαγραμμένου αρχείου (μη τεκμηριωμένη). CF=1 εάν όχι επιτυχής AX = κωδικός σφάλματος.

Σημείωση: Το DOS δεν διαγράφει τα δεδομένα ενός αρχείου, απλώς αυτά γίνονται μη προσπελάσιμα λόγω ενημέρωσης του FAT. Διαγραφή ενός ανοικτού αρχείου μπορεί να προκαλέσει βλάβη στο σύστημα αρχείων.

πχ.
```
; delete file c:\emu8086\vdrive\C\test1\file2.txt
mov ah, 41h
mov dx, offset file2
int 21h
```

Διαγραφή Καταλόγου (διεύθυνση-διαδρομή):

- **INT 21h / AH= 3Ah**

Είσοδος: DS:DX -> Το path (σε ASCII) του καταλόγου προς διαγραφή.

Επιστρέφει: CF=0 εάν διαγραφεί επιτυχώς, CF=1 εάν όχι & AX = κωδικός σφάλματος.

Σημείωση: ο κατάλογος πρέπει να είναι κενός (χωρίς αρχεία στο εσωτερικό του).

πχ. ; delete directory: c:\emu8086\MyBuild\dir1

```
mov ah, 3ah
mov dx, offset dir1
int 21h
```

Λειτουργία SEEK
(Ορίζει την τρέχουσα θέση αρχείου) :

- **INT 21h / AH= 42h**

Είσοδος: AL = αφετηρία της μετατόπισης: 0 – αρχή του αρχείου. 1 – τρέχουσα θέση αρχείου. 2 – τέλος αρχείου. BX = file handle. CX:DX = μετατόπιση από την αρχική θέση αρχείου. Επιστρέφει: CF=0 εάν επιτυχής,

DX:AX = νέα θέση αρχείου σε bytes από την αρχή του αρχείου. CF=1 εάν όχι επιτυχής AX = κωδικός σφάλματος.

Σημείωση: για αφετηρία μετατόπισης 1 και 2, ο δείκτης μπορεί να τοποθετηθεί πριν την αρχή του αρχείου. Στην περίπτωση αυτή δεν υπάρχει σφάλμα αλλά θα υπάρξει σφάλμα αν κατόπιν γίνουν απόπειρες εγγραφής ή ανάγνωσης του αρχείου. Εάν η νέα θέση είναι πέρα από το τρέχον τέλος του αρχείου, το αρχείο θα επεκταθεί με την επόμενη εγγραφή.

Λειτουργία SEEK
(Ορίζει την τρέχουσα θέση αρχείου) :

πχ1.

```
; seek:
mov ah, 42h
mov bx, handle
mov al, 0
mov cx, 0
mov dx, 10 ; metatopisi 10 byte
int 21h
```

πχ2.

```
; seek:
mov ah, 42h
mov bx, handle
mov al, 0
mov cx, 0
mov dx, 2 ; metatopisi  2 byte
int 21h
```

Παράδειγμα (1/3)

```
name "file"
org  100h

jmp start

dir1 db "c:\test1", 0
dir2 db "test2", 0
dir3 db "newname", 0
file1 db "c:\test1\file1.txt", 0
file2 db "c:\test1\newfile.txt", 0
file3 db "t1.txt", 0
handle dw ?

text db "lazy dog jumps over red fox."
text_size = $ - offset text
text2 db "hi!"
text2_size = $ - offset text2

start:
mov ax, cs
mov dx, ax
mov es, ax
; create c:\emu8086\vdrive\C\test1
mov dx, offset dir1
mov ah, 39h
int 21h
; create  c:\emu8086\MyBuild\test2
mov dx, offset dir2
mov ah, 39h
int 21h
; rename directory: c:\emu8086\MyBuild\test2 to c:\emu8086\MyBuild\newname
mov ah, 56h
mov dx, offset dir2   ; existing.
mov di, offset dir3   ; new.
int 21h
; create and open file: c:\emu8086\vdrive\C\test1\file1.txt
mov ah, 3ch
mov cx, 0
mov dx, offset file1
int 21h
jc err
mov handle, ax
; write to file:
mov ah, 40h
mov bx, handle
mov dx, offset text
mov cx, text_size
int 21h
```

Παράδειγμα (2/3)

```
; close c:\emu8086\vdrive\C\test1\file1.txt
mov ah, 3eh
mov bx, handle
int 21h
err:
nop
; rename fileL c:\emu8086\vdrive\C\test1\file1.txt to c:\test1\newfile.txt
mov ah, 56h
mov dx, offset file1    ; existing.
mov di, offset file2    ; new.
int 21h
; delete file c:\emu8086\vdrive\C\test1\newfile.txt
mov ah, 41h
mov dx, offset file2
int 21h
; delete directory: c:\emu8086\vdrive\C\test1
mov ah, 3ah
mov dx, offset dir1
int 21h
; create and open file: c:\emu8086\MyBuild\t1.txt
mov ah, 3ch
mov cx, 0
mov dx, offset file3
int 21h
jc err2
mov handle, ax
; seek:
mov ah, 42h
mov bx, handle
mov al, 0
mov cx, 0
mov dx, 10
int 21h
; write to file:
mov ah, 40h
mov bx, handle
mov dx, offset text
mov cx, text_size
int 21h
; seek:
mov ah, 42h
mov bx, handle
mov al, 0
mov cx, 0
mov dx, 2
int 21h
```

Παράδειγμα (3/3)

```
; write to file:
mov ah, 40h
mov bx, handle
mov dx, offset text2
mov cx, text2_size
int 21h
; close c:\emu8086\MyBuild\t1.txt
mov ah, 3eh
mov bx, handle
int 21h
err2:
nop
; delete file  c:\emu8086\MyBuild\t1.txt
mov ah, 41h
mov dx, offset file3
int 21h
; delete directory: c:\emu8086\MyBuild\newname
mov ah, 3ah
mov dx, offset dir3
int 21h

ret
```

- Το παράδειγμα που προηγήθηκε βρίσκεται στα examples του emu8086 με όνομα file-operations.

Παράδειγμα ανάγνωσης Αρχείου (1/4)

Έστω έχουμε δηλώσει στο data segment μια περιοχή προσωρινής μνήμης με όνομα buffer μεγέθους 100, στην οποία θα αποθηκεύσουμε τα δεδομένα που θα διαβάσουμε από ένα αρχείο. Το όνομα του αρχείου έχει οριστεί στο τμήμα δεδομένων με τελευταίο χαρακτήρα το 0 (ονομάζεται 'null terminated string'). Π.χ.

```
handle dw ?
buffer db 100 dup(0)
file1 db "c:\test1\file1.txt",0
```

για να διαβάσουμε από αυτό το αρχείο (που υπάρχει) θα πρέπει να το ανοίξουμε για ανάγνωση [π.χ. read-only που ενεργοποιείται αν θέσουμε στο AL την τιμή 0, να δώσουμε στο AH την τιμή 3Dh (άνοιγμα αρχείου), να τοποθετήσουμε τη διεύθυνση του ονόματος στο DX (αν έχουμε τροποποιήσει το DS και δε δείχνει στο τμήμα δεδομένων, θα πρέπει να τον επαναφέρουμε), και να ανοίξουμε το αρχείο. Θα πρέπει το file handle που θα δημιουργηθεί να τοποθετηθεί σε μια θέση μνήμης, π.χ. handle. Το handle είναι ο δείκτης μέσω του οποίου θα διαβάσουμε ή θα γράψουμε. Αν δε το διατηρήσουμε, τότε δε θα μπορούμε να έχουμε πρόσβαση στο αρχείο.

Παράδειγμα ανάγνωσης Αρχείου (2/4)

```
;open readonly

mov ah,3dh                  ; call the 'open Existing
File'
mov al,0                    ; open the file with READ
mode
mov dx,offset file1         ; load the memory address to
DX
int 21h                     ; call int to open the file
mov handle,ax               ; save handle to memory
JC errorfileopen            ; if CF==1 then an error has
                            ; occured.Go to error block
```

Παράδειγμα ανάγνωσης Αρχείου (3/4)

Υποθέτουμε ότι έχουμε ανοίξει το αρχείο ή δημιουργήσει με δυνατότητα ανάγνωσης. Ο handle έχει αποθηκευτεί στη θέση handle του Data Segment.

Έχουμε δηλώσει τη προσωρινή θέση μνήμης buffer, όπως στη προηγούμενη διαφάνεια.

Θέτουμε το 3FH στο AH για την ανάγνωση αρχείου.

Μεταφέρουμε το handle στο BX

Φορτώνουμε τη διεύθυνση του buffer στο DX. (αν έχουμε τροποποιήσει το DS και δε δείχνει στο τμήμα δεδομένων, θα πρέπει να τον επαναφέρουμε). Θέτουμε στο CX τον αριθμό των Byte που θέλουμε να διαβάσουμε. Η ανάγνωση ξεκινάει από τη θέση που είναι ο file cursor. Αν δεν έχουμε διαβάσει καθόλου και μόλις έχουμε ανοίξει το αρχείο ο file cursor είναι στη θέση 0. Αν διαβάσουμε 3 Byte, τότε ο file cursor θα δείχνει 3, αφού θα έχουν διαβαστεί τα Byte 0,1,2. Αν διαβάσουμε πάλι 3 Byte, τότε θα διαβαστούν τα 3,4,5 και ο file cursor θα δείχνει στη θέση 6. Μπορούμε με τη λειτουργία seek να τοποθετήσουμε το cursor σε όποιο byte θέλουμε.

Παράδειγμα ανάγνωσης Αρχείου (4/4)

```
;read data from file

MOV AH,3FH                  ; call the 'Read File'
MOV BX,handle               ; copy the handle to BX
LEA DX,buffer               ; find the buffer address and place
it on DX
MOV CX,03                   ; read 3 bytes
INT 21H                     ; call int to perform its duty
JC errorfileread            ; if CF==1 then an error has
                            ; occured. Go to error block
```

Τέλος Ενότητας

Με τη συγχρηματοδότηση της Ελλάδας και της Ευρωπαϊκής Ένωσης