Αρχιτεκτονική Υπολογιστών

Ενότητα 11: Γραφικά VGA

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.

Ρύθμιση του τρόπου λειτουργίας της οθόνης

- Η οθόνη έχει 2 τρόπους λειτουργίας:
  - Τρόπος λειτουργίας κειμένου.
  - Τρόπος λειτουργίας γραφικών.

- Προκειμένου να ρυθμίσουμε τον τρόπο λειτουργίας της οθόνης για γραφικά χρησιμοποιούμε το int 10h.
  - Ο καταχωρητής ah=0 *(ρύθμιση οθόνης).*
  - Ο καταχωρητής al έχει την τιμή που αντιστοιχεί σε ένα συγκεκριμένο ρυθμό λειτουργίας.

Ρύθμιση λειτουργίας της οθόνης

- Υποστηριζόμενοι ρυθμοί λειτουργίας emu8086:
  - 00h - **text mode**. 40x25. 16 colors. 8 pages.
  - 03h - **text mode**. 80x25. 16 colors. 8 pages.
  - 13h - **graphical mode**. 40x25. 256 colors. 320x200 pixels. 1 page.
- Υπάρχουν και άλλοι τρόποι λειτουργίας που **ΔΕΝ** υποστηρίζονται από το emu8086.
- Παράδειγμα ρύθμισης οθόνης για γραφικά:

```
mov al, 13h
mov ah, 0
int 10h
```

Τοποθέτηση pixel (1/2)

- Στην κατάσταση γραφικών μπορούμε να τοποθετήσουμε pixels σε οποιοδήποτε σημείο θέλουμε.
- Τα pixel έχουν μια τιμή από 0-255 *(Byte).*
- Η πρόσβαση στην οθόνη γίνεται με την εγγραφή στη μνήμη γραφικών η οποία ξεκινάει από τη θέση 0A000h. Δηλαδή στη θέση A000h αντιστοιχεί η πάνω αριστερή γωνία της οθόνης.
- Προκειμένου να μετατρέψουμε τις συντεταγμένες (x,y) στην αντίστοιχη μετατόπιση της διεύθυνσης μνήμης από το A000 *(για μια οθόνης 320x200 pixels)* κάνουμε την πράξη y*320+x.

Τοποθέτηση pixel (2/2)

- Για παράδειγμα αν θέλουμε να τοποθετήσουμε ένα pixel με τιμή χρώματος 9 *(bright-blue)* στις συντεταγμένες (x,y) θα πρέπει με την εντολή mov να τοποθετήσουμε την τιμή 9 στη θέση A000 με μετατόπιση την τιμή που προκύπτει από τη συνάρτηση y*320+x.

- Υπάρχει μια παλέτα 255 χρωμάτων. Η παλέτα είναι προκαθορισμένη και αντιστοιχεί στον τρόπο λειτουργίας MCGA 320x200 pixels.

Παράδειγμα τοποθέτησης pixel

- Αν θέλουμε να τοποθετήσουμε ένα pixel στις συντεταγμένες (x,y) τότε θα πρέπει να κάνουμε τα όλα τα παρακάτω βήματα:
- Πολλαπλασιασμό y*320:

```
mov ax,y
mov cx,320
mul cx
```

- Πρόσθεση του x:

```
add ax,x
```

Βελτιστοποίηση πολλαπλασιασμού *(τοποθέτησης pixel)*

- Η πράξη του πολλαπλασιασμού πρέπει να αποφεύγεται ΑΝ ΜΠΟΡΟΥΜΕ γιατί:
  - Καθυστερεί πολύ να ολοκληρωθεί.
  - Καθυστερεί όλη τη διοχέτευση.
  - Για 2Byte απαιτεί 2 καταχωρητές (π.χ. AX,CX) για τον πολλαπλασιασμό και τροποποιεί 2 καταχωρητές (DX,AX).

- Υπάρχει περίπτωση να απομακρύνουμε τον πολλαπλασιασμό με εντολές ολίσθησης και πρόσθεσης ΑΝ θέλουμε να πολλαπλασιάσουμε με μια σταθερή τιμή που μπορεί να εκφραστεί με δυνάμεις του 2.

Βελτιστοποίηση πολλαπλασιασμού

- Η πράξη 320*y+x μπορεί να εκφραστεί ως:
  - 64*y + 256*y +x .
- Απαιτούνται λοιπόν:
  - 6 ολισθήσεις προς αριστερά του y.
  - 8 ολισθήσεις προς αριστερά του αρχικού y.
  - Πρόσθεση των 2 παραπάνω.
  - Πρόσθεση του x.

```
mov AX,y
shl AX,6
mov BX,y
shl BX,8
add ax,bx
add ax,x
```

Παράδειγμα τοποθέτησης pixel
(συνέχεια…)

- Εγγραφή στην αντίστοιχη μετατόπιση ξεκινώντας από το A000. Για να γίνει αυτό τοποθετούμε σε έναν καταχωρητή τμήματος την τιμή A000 και στη συνέχεια χρησιμοποιούμε τη διευθυνσιοδότηση μετατόπισης.
  - Τοποθέτηση της A000 σε καταχωρητή τμήματος:

```
mov bx,0A000h
mov es,bx
```

  - Εγγραφή με διευθυνσιοδότηση μετατόπισης:

```
mov si,ax
mov es:[si], 9
```

Η εντολή xchg

- Η εντολή xchg η οποία κάνει αμοιβαία ανταλλαγή δύο καταχωρητών.
- Υπάρχουν τρεις παραλλαγές:

```
XCHG reg, reg
XCHG reg, mem
XCHG mem, reg
```

Παραδείγματα:

```
xchg    ax, bx        ; Put AX in BX and BX in AX
xchg    memory, ax    ; Put "memory" in AX and AX in "memory"
xchg    mem1, mem2    ; Illegal, can't exchange memory locations!
```

Βελτιστοποίηση αλλαγής τιμής καταχωρητών

- Η παρακάτω εντολή τοποθετεί στο SI την τιμή του AX.

```
mov si,ax
```

- Μια πιο γρήγορη υλοποίηση είναι η εντολή xchg.
- Αυτό γίνεται ως εξής:

```
xchg si,ax
```

Σχεδιασμός οριζόντιας γραμμής

- Προκειμένου να σχεδιάσουμε μια οριζόντια γραμμή προς τα δεξιά, αρκεί να αυξήσουμε το δείκτη μνήμης κατά 1.
- Δηλαδή να επαναληφθεί για όσα pixel μήκος θέλουμε το:

```
mov es:[si],9
inc si
```

- Για να σχεδιάσουμε μια οριζόντια γραμμή προς τα αριστερά, αρκεί να αφαιρούμε κάθε φορά 1.

Σχεδιασμός κατακόρυφης γραμμής

- Προκειμένου να σχεδιάσουμε μια κατακόρυφη γραμμή προς τα κάτω, αρκεί να αυξήσουμε το δείκτη μνήμης κατά τα pixel μιας γραμμής, δηλαδή κατά 320.
- Δηλαδή να επαναληφθεί για όσα pixel μήκος θέλουμε το:

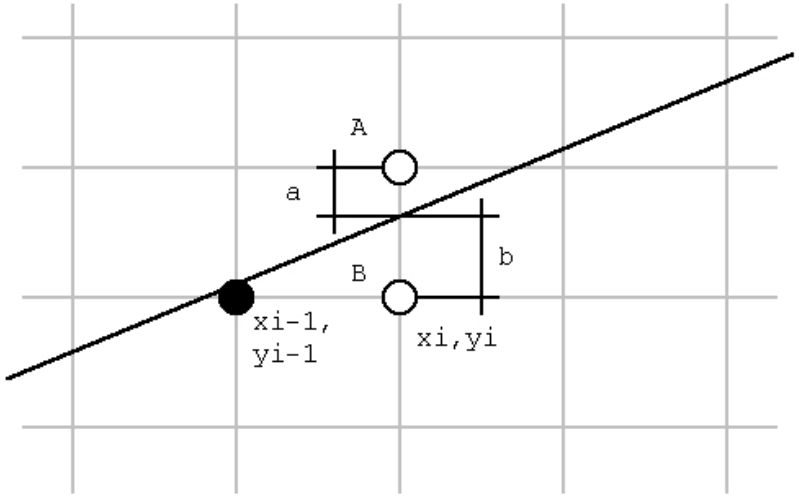
```
mov es:[si],9
add si,320
```

- Για να σχεδιάσουμε μια κατακόρυφη γραμμή προς την κορυφή, αρκεί να αφαιρούμε κάθε φορά 320.

Σχεδιασμός διαγώνιας γραμμής (1/2)

- Είναι αρκετά δύσκολο να σχεδιάσουμε μια διαγώνια γραμμή.
- Δεν έχουμε στη διάθεσή μας συναρτήσεις cos, sqrt, sin κτλ.
- Πρέπει να χρησιμοποιούμε το mul όσο λιγότερο γίνεται.
- Η εξίσωση της γραμμής είναι: $y = \frac{dy}{dx}$
- Ο πιο γνωστός αλγόριθμος σχεδιασμού γραμμής σε οθόνη raster *(με pixel ή διακριτές θέσεις)* είναι ο αλγόριθμος Bresenham.
- Χρησιμοποιείται ευρέως γιατί υλοποιείται μόνο με πράξεις πρόσθεσης, αφαίρεσης και ολίσθησης.

Σχεδιασμός διαγώνιας γραμμής (2/2)



- Επειδή τα pixel είναι διακριτά στην οθόνη, θα πρέπει να υπολογίζουμε κάθε φορά αν θα σχεδιάσουμε είτε το pixel A είτε το pixel B, αναλόγως του ποιου έχει την κοντινότερη απόσταση προς τη γραμμή.
- http:// en.wikipedia.org/wiki/Bresenham's_line_algorithm

Σχεδιασμός κύκλου (1/3)

- **Ο σχεδιασμός κύκλου είναι αρκετά δύσκολος γιατί απαιτεί τριγωνομετρικές εξισώσεις.**
- **Μπορεί να απλοποιηθεί εν μέρη με την εξής παραδοχές:**
  - Το κάτω ημικύκλιο είναι συμμετρικό του άνω. Άρα αρκεί να υπολογίζουμε τα (x,y) στο άνω ημικύκλιο και στη συνέχεια υπολογίζουμε τα συμμετρικά σημεία.
  - Το άνω αριστερό τεταρτημόριο είναι συμμετρικό του άνω δεξιού τεταρτημόριου.
  - Οι πρώτες 45 μοίρες ενός τεταρτημόριου είναι συμμετρικές ως προς τις υπόλοιπες 45 του τεταρτημόριου.
- **Απαιτείται λοιπόν ο υπολογισμός των σημείων των πρώτων 45 μοιρών και εύκολα υπολογίζονται τα υπόλοιπα σημεία.**

Σχεδιασμός κύκλου (2/3)

- Για κάθε σημείο που υπολογίσουμε στις πρώτες 45 μοίρες τοποθετούμε 8 pixels στην οθόνη, σύμφωνα με τις εξισώσεις συμμετρίας.
- Ο ψευδοκώδικας είναι:

start at pixel(w,0)

line02:

mirror and plot for 8octants

if h = w then end

move up 1 pixel

if outside circle then move left 1 pixel

repeat from line02

Σχεδιασμός κύκλου (3/3)

Ψευδοκώδικας

- **r = 100 : x = 160 : y = 100 : c = 1**
- **w = r : h = 0**
- **d = w * w + h * h - r * r**
- **DO**

  - PIXELSET(x + w, y + h),c : PIXELSET(x + h, y + w),c
  - PIXELSET(x + w, y - h),c : PIXELSET(x + h, y - w),c
  - PIXELSSET(x - w, y + h),c : PIXELSET(x - h, y + w),c
  - PIXELSET(x - w, y - h),c : PIXELSET(x - h, y - w),c

  IF h = w THEN END

  - d = d + h + h + 1
  - h = h + 1

  IF d > 0 THEN

  - d = d - w - w - 1
  - w = w - 1

  END IF
  LOOP

Ο κώδικας ASSEMBLY που υλοποιεί το σχεδιασμό κύκλου βρίσκεται στο google αν βάλετε search string: “how-can-i-make-a-circle-in-assembly”

Ασύγχρονη είσοδος με int 16h και χρήση πλήκτρων χωρίς ASCII τιμή (1/8)

- Η είσοδος χαρακτήρα από το πληκτρολόγιο με int 21h είναι η πιο συχνά χρησιμοποιούμενη μέθοδος εισαγωγής δεδομένων.
- Είναι μια σύγχρονη μέθοδος εισόδου, που σημαίνει ότι ο επεξεργαστής σταματάει την εκτέλεση του προγράμματος *(περιμένει)* το χρήστη.
- Υπάρχουν περιπτώσεις που θέλουμε να συνεχίζει η εκτέλεση αν δεν έχει πατήσει ο χρήστης κάποιο πλήκτρο. Για παράδειγμα θέλουμε αν καθώς γίνεται η επεξεργασία ο χρήστης πατήσει το πλήκτρο ESC τότε να σταματήσει το πρόγραμμα.

Ασύγχρονη είσοδος με int 16h και χρήση πλήκτρων χωρίς ASCII τιμή (2/8)

- Επειδή αυτό είναι ένα ασύγχρονο γεγονός, θα πρέπει να το υλοποιήσουμε με ένα διαφορετικό int και αυτό είναι το int 16h
- Επίσης το int 16h μας δίνει μια δυνατότητα που δεν υπάρχει στο int 21h. Αυτή είναι η δυνατότητα να διαβάσουμε πλήκτρα που δεν αντιστοιχούν στον κώδικα ASCII, όπως τα βελάκια, τα function keys ή τα κουμπιά ήχου, e-mail κ.α. που βρίσκονται σε κάποια πληκτρολόγια.
- Τα ειδικά αυτά κουμπιά δεν έχουν τιμή ASCII.
- Όλα τα πλήκτρα *(είτε έχουν ASCII τιμή είτε όχι)* έχουν μια τιμή **Bios Scan code.**

Ασύγχρονη είσοδος με int 16h και χρήση πλήκτρων χωρίς ASCII τιμή (3/8)

- Για να κάνουμε ασύγχρονο έλεγχο θα πρέπει να κατασκευάσουμε μια συνάρτηση η οποία θα δίνει τιμή AH=01 και θα καλεί το int 16h.
- Αυτό έχει ως συνέπεια τον έλεγχο αν έχει πατηθεί ένα πλήκτρο:

  INT 16h / AH = 01h - check for keystroke in the keyboard buffer.
- Οι τιμές που επιστρέφονται είναι:

  ZF = 1 if keystroke is not available.

  ZF = 0 if keystroke available.

  AH = BIOS scan code.

  AL = ASCII character.

Ασύγχρονη είσοδος με int 16h και χρήση πλήκτρων χωρίς ASCII τιμή (4/8)

- Αμέσως μετά ελέγχουμε το Zero Flag (**ZF**).
- Ο έλεγχος μπορεί να γίνει με μια εντολή JNE ή JE.
- Αυτό ισχύει γιατί αυτές οι εντολές διακλάδωσης ελέγχουν το ZF. Η εντολή JE ενεργοποιείται όταν έχουμε ισότητα, δηλαδή όταν ZF=1. Αντιστρόφως, η εντολή JNE ενεργοποιείται όταν δεν έχουμε ισότητα, δηλαδή όταν ZF=0.
- Αν το ZF=0 τότε έχει πατηθεί χαρακτήρας.
- Αν το ZF=1 δεν έχει πατηθεί χαρακτήρας.

Ασύγχρονη είσοδος με int 16h και χρήση πλήκτρων χωρίς ASCII τιμή (5/8)

- Αν έχει πατηθεί χαρακτήρας τότε ο χαρακτήρας θα βρίσκεται στο AH AL.
- Αν το AL δεν είναι 0, τότε το AL έχει την ASCII τιμή του χαρακτήρα που πατήθηκε.
- Αν το AL=0 τότε έχουμε ειδικό πλήκτρο και στο AH κοιτάζουμε το bios scan code.
- **Προσοχή**: Ο χαρακτήρας παραμένει στο keyboard buffer. Συνιστάται να χρησιμοποιήσουμε μια κλήση int 16h, όπως στην επόμενη διαφάνεια για να απομακρυνθεί.
- Αν δεν απομακρυνθεί ο χαρακτήρας τότε θα φαίνεται ότι ο χρήστης πατάει συνέχεια τον ίδιο χαρακτήρα.

Ασύγχρονη είσοδος με int 16h και χρήση πλήκτρων χωρίς ASCII τιμή (6/8)

- Θα πρέπει ΑΝ υπάρχει χαρακτήρας (δηλαδή ZF=0) να τοποθετήσουμε ah=00, int 16h, για να διαβάσουμε τον χαρακτήρα και να απομακρυνθεί από το keyboard buffer.
- Αν το AL δεν είναι 0, τότε το AL έχει την ASCII τιμή του χαρακτήρα που πατήθηκε.
- Αν το AL=0 τότε έχουμε ειδικό πλήκτρο και στο AH κοιτάζουμε το bios scan code.
- Υπάρχουν λίστες με τους bios scan codes για κάθε είδους πληκτρολογίου (π.χ. Πληκτρολόγιο XT, AT, MF, 102 πλήκτρων, 84 πλήκτρων κτλ).
- Για παράδειγμα το βελάκι δεξιά έχει scan code 4Dh ενώ το βελάκι αριστερά 4Bh.

Ασύγχρονη είσοδος με int 16h και χρήση πλήκτρων χωρίς ASCII τιμή (7/8)

- Η συνάρτηση μας λοιπόν αν δεν έχει πατηθεί κάποιο πλήκτρο δε θα κάνει τίποτα παραπάνω και θα πηγαίνει στο RET.
- Αν έχει πατηθεί κάποιο πλήκτρο θα επεξεργάζεται κατάλληλα. Για παράδειγμα αν είναι το πλήκτρο εξόδου που ορίσαμε 'q' τότε θα εκτελείται το mov ah,4ch, int 21h που τερματίζει το πρόγραμμα.
- Θα πρέπει να τοποθετήσουμε την κλήση της συνάρτηση μας σε διάφορα σημεία που να εκτελείται κατά διαστήματα. Καλές επιλογές είναι μέσα στους βρόχους ή μέσα σε συναρτήσεις που εκτελούνται πολύ συχνά.

Ασύγχρονη είσοδος με int 16h και χρήση πλήκτρων χωρίς ASCII τιμή (8/8)

- Η συνάρτηση που κατασκευάσαμε εδώ χρησιμοποιεί την τεχνική polling για εισαγωγή πλήκτρου.
- Δεν είναι η βέλτιστη συνάρτηση εισόδου, γιατί ο επεξεργαστής κάθε φορά ελέγχει αν υπάρχει κάποιο πλήκτρο, δηλαδή εκτελεί εντολές που μπορεί να μη χρειάζονται.
- Υπάρχει καλύτερη υλοποίηση *(και πολύ πιο δύσκολη).*

Τέλος Ενότητας

Με τη συγχρηματοδότηση της Ελλάδας και της Ευρωπαϊκής Ένωσης