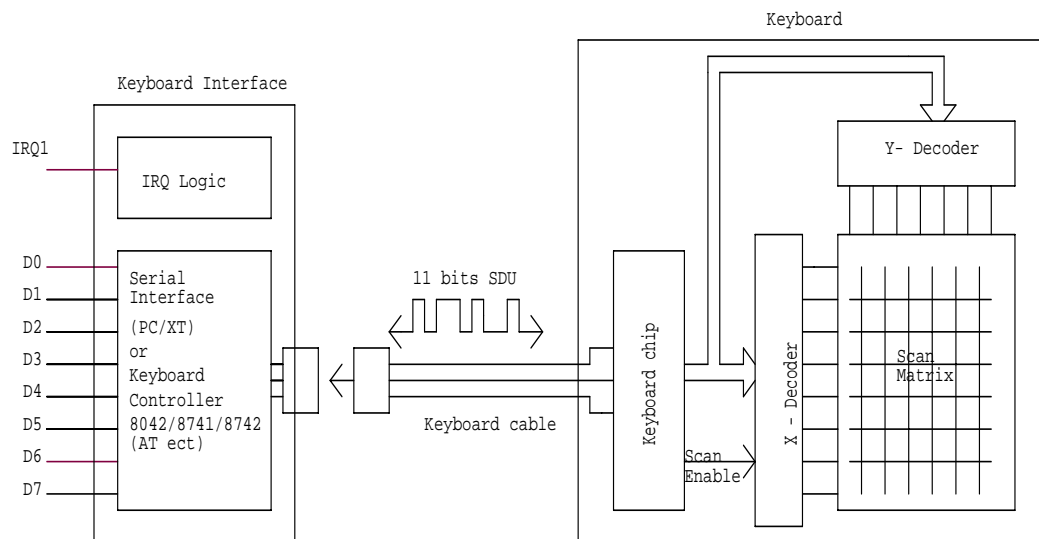


Chương 3 GIAO TIẾP THIẾT BỊ CHUẨN

1. Giao tiếp bàn phím

1.1. Nguyên lý hoạt động



Hình 3.1 - Sơ đồ nguyên lý và các ghép nối của bàn phím

Chip xử lý bàn phím liên tục kiểm tra trạng thái của ma trận quét (scan matrix) để xác định công tắc tại các tọa độ X, Y đang được đóng hay mở và ghi một mã tương ứng vào bộ đệm bên trong bàn phím. Sau đó mã này sẽ được truyền nối tiếp tới mạch ghép nối bàn phím trong PC. Cấu trúc của SDU (Serial Data Unit) cho việc truyền số liệu:

0										10
STRT	DB ₀	DB ₁	DB ₂	DB ₃	DB ₄	DB ₅	DB ₆	DB ₇	PAR	STOP

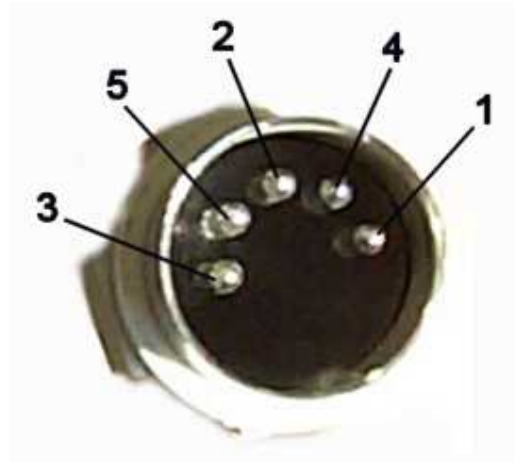
STRT: bit start (luôn bằng 0)

DB₀ - DB₇: bit số liệu từ 0 đến 7.

PAR: bit parity (luôn lẻ)

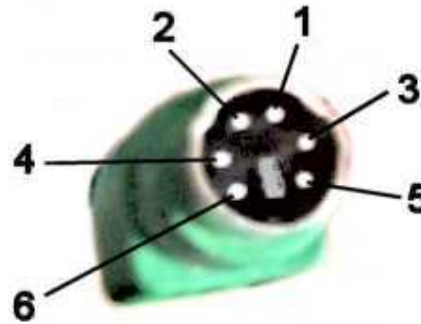
STOP: bit stop (luôn bằng 1).

- Chân 1: clock
- Chân 2: dữ liệu
- Chân 3: Reset
- Chân 4: GND
- Chân 5: Vcc



Hình 3.2 – Đầu cắm bàn phím AT

- Chân 1: dữ liệu
- Chân 2: không dùng
- Chân 3: GND
- Chân 4: Vcc
- Chân 5: clock
- Chân 6: không dùng



Hình 3.3 – Đầu cắm bàn phím PS/2

Mỗi phím nhấn sẽ được gán cho 1 mã quét (scan code) gồm 1 byte. Nếu 1 phím được nhấn thì bàn phím phát ra 1 mã make code tương ứng với mã quét truyền tới mạch ghép nối bàn phím của PC. Ngắt cứng INT 09h được phát ra qua IRQ1.

Chương trình xử lý ngắt sẽ xử lý mã này tùy theo phím SHIFT có được nhấn hay không. Ví dụ: nhấn phím SHIFT trước, không rời tay và sau đó nhấn ‘C’:

make code được truyền - 42(SHIFT) - 46 (‘C’).

Nếu rời tay nhấn phím SHIFT thì bàn phím sẽ phát ra break code và mã này được truyền như make code. Mã này giống như mã quét nhưng bit 7 được đặt lên 1, do vậy nó tương đương với make code cộng với 128. Tùy theo break code, chương trình con xử lý ngắt sẽ xác định trạng thái nhấn hay rời của các phím. Thí dụ, phím SHIFT và ‘C’ được rời theo thứ tự ngược lại với thí dụ trên:

break code được truyền 174 (bằng 46 cộng 128 tương ứng với ‘C’) và 170 (bằng 42 cộng 128 tương ứng với SHIFT).

Phần cứng và phần mềm xử lý bàn phím còn giải quyết các vấn đề vật lý sau:

- Nhấn và nhả phím nhưng không được phát hiện.

- Khử nhiễu rung cơ khí và phân biệt 1 phím được nhấn nhiều lần hay được nhấn chỉ 1 lần nhưng được giữ trong một khoảng thời gian dài.

1.2. Lập trình giao tiếp qua các cổng

Bàn phím cũng là một thiết bị ngoại vi nên về nguyên tắc có thể truy xuất nó qua các cổng vào ra.

❖ Các thanh ghi và các port:

Sử dụng 2 địa chỉ port 60h và 64h có thể truy xuất bộ đệm vào, bộ đệm ra và thanh ghi điều khiển của bàn phím.

Port	Thanh ghi	R/W
60h	Đệm ngõ ra	R
60h	Đệm ngõ vào	W
64h	Thanh ghi điều khiển	W
64h	Thanh ghi trạng thái	R

Thanh ghi trạng thái xác định trạng thái hiện tại của bộ điều khiển bàn phím. Thanh ghi này chỉ đọc (read only) và đọc bằng lệnh IN tại port 64h.

7							0
PARE	TIM	AUXB	KEYL	C/D	SYSF	INPB	OUTB

PARE: Lỗi chặn lẻ của byte cuối cùng được vào từ bàn phím; 1 = có lỗi chặn lẻ, 0 = không có.

TIM: Lỗi quá thời gian (time-out); 1 = có lỗi, 0 = không có.

AUXB: Đệm ra cho thiết bị phụ (chỉ có ở máy PS/2); 1 = giữ số liệu cho thiết bị, 0 = giữ số liệu cho bàn phím.

KEYL: Trạng thái khóa bàn phím; 1 = không khóa, 0 = khóa.

C/D: Lệnh/dữ liệu; 1 = Ghi qua port 64h, 0 = Ghi qua port 60h.

SYSF: cờ hệ thống; 1 = tự kiểm tra thành công, 0 = reset khi cấp điện

INPB: Trạng thái đệm vào; 1 = dữ liệu CPU trong bộ đệm vào, 0 = đệm vào rỗng.

OUTB: Trạng thái đệm ra; 1 = dữ liệu bộ điều khiển bàn phím trong bộ đệm ra, 0 = đệm ra rỗng.

Thanh ghi điều khiển

Các lệnh cho bộ điều khiển bàn phím:

Mã	Mô tả
A7h	Cấm thiết bị phụ
A8h	Cho phép thiết bị phụ
A9h	Kiểm tra giao tiếp thiết bị phụ và lưu mã kiểm tra vào bộ đệm ra 00h: không lỗi 01h: CLK ở mức thấp 02h: CLK ở mức cao 03h: DATA ở mức thấp

	04h: DATA ở mức cao FFh: lỗi khác
AAh	Tự kiểm tra (ghi 55h vào bộ đệm ra nếu không lỗi)
ABh	Kiểm tra giao tiếp bàn phím và lưu mã kiểm tra vào bộ đệm ra
ADh	Cắm bàn phím
A Eh	Cho phép bàn phím
C0h	Đọc cổng vào và truyền dữ liệu đến bộ đệm ra
C1h	Đọc các bit 3 – 0 của cổng vào và truyền đến các bit 3- 0 của thanh ghi trạng thái cho đến khi INPB = 1
C2h	Đọc các bit 7 – 4 của cổng vào và truyền đến các bit 7- 4 của thanh ghi trạng thái cho đến khi INPB = 1
D0h	Đọc cổng ra
D1h	Ghi cổng ra
D2h	Ghi vào bộ đệm ra và xoá AUXB
D3h	Ghi vào bộ đệm ra và set AUXB
D4h	Ghi byte dữ liệu tiếp theo vào thiết bị phụ

Khóa bàn phím:

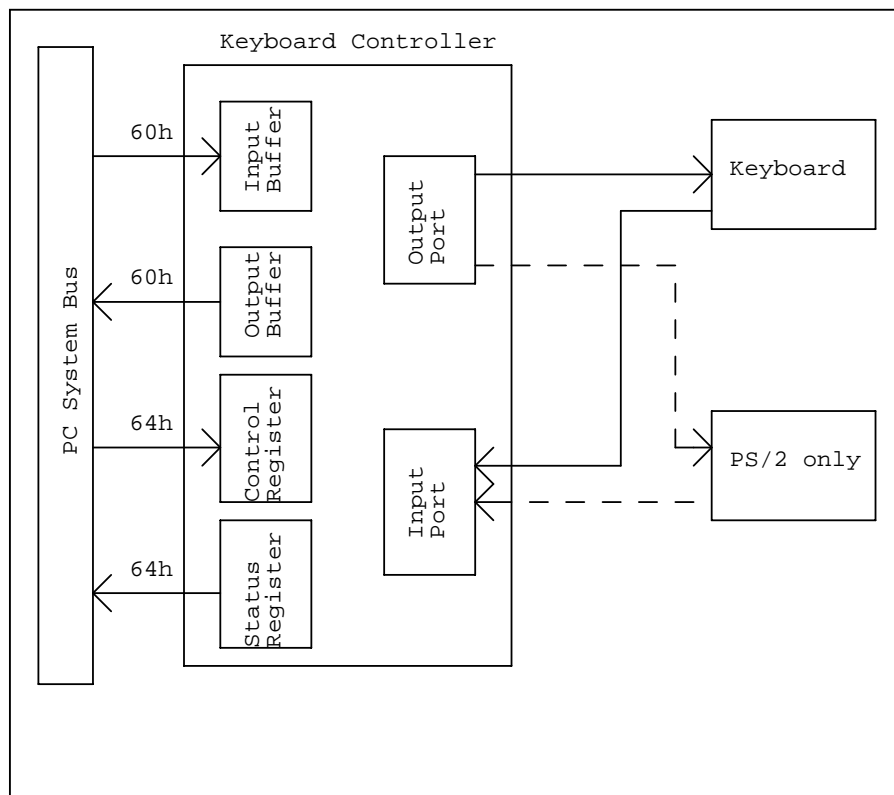
Start:

IN AL, 64h ; đọc byte trạng thái

TEST AL, 02h ; kiểm tra bộ đệm có đầy hay không

JNZ start

OUT 64h, 0ADh ; khóa bàn phím



Hình 3.4 - Bộ điều khiển bàn phím

❖ **Các lệnh cho bàn phím:**

Mã	Lệnh	Mô tả
EDh	Bật/tắt LED	Bật/tắt các đèn led của bàn phím
EEh	Echo	Trả về byte Eeh
F0h	Đặt/nhận dạng mã quét	Đặt 1 trong 3 tập mã quét và nhận diện các mã quét tập mã quét hiện tại.
F2h	Nhận diện bàn phím	Nhận diện ACK = AT, ACK+abh+41h=MF II.
F3h	Đặt tốc độ lặp lại/trễ	Đặt tốc độ lặp lại và thời gian trễ của bàn phím
F4h	Enable	Cho phép bàn phím hoạt động
F5h	Chuẩn/không cho phép	Đặt giá trị chuẩn và cấm bàn phím.
F6h	Chuẩn/cho phép	Đặt giá trị chuẩn và cho phép bàn phím.
FEh	Resend	Bàn phím truyền ký tự cuối cùng một lần nữa tới bộ điều khiển bàn phím
FFh	Reset	Chạy reset bên trong bàn phím

Thí dụ: lệnh bật đèn led cho phím NUMCLOCK, tắt tất cả các đèn khác.

```
MOV AL,0EDh
OUT 60H, AL
WAIT:
IN AL, 64H          ; đọc thanh ghi trạng thái
JNZ WAIT
MOV AL,02h
OUT 60H, AL          ; bật đèn cho numclock
```

Cấu trúc của byte chỉ thị như sau:

7	6	5	4	3	2	1	0
0	0	0	0	0	CPL	NUM	SCR

CPL: 1 = bật đèn Caps Lock; 0 = tắt

NUM: 1 = bật đèn Num Lock; 0 = tắt

SCR: 1 = bật đèn Scroll Lock; 0 = tắt

1.3. Lập trình giao tiếp qua các hàm của DOS, BIOS

BIOS ghi các ký tự do việc nhấn các phím vào bộ đệm tạm thời được gọi là bộ đệm bàn phím (keyboard buffer), có địa chỉ 40h:1Eh, gồm 32 byte và kết thúc ở địa chỉ 40h:3Dh. Mỗi ký tự được lưu trữ bằng 2 byte, byte cao là mã quét, và byte thấp là mã ASCII. Chương trình xử lý ngắt sẽ xác định mã ASCII từ mã quét bằng bảng biến đổi và ghi cả 2 mã vào bộ đệm bàn phím. Bộ đệm bàn phím được tổ chức như bộ đệm vòng (ring buffer) và được quản lý bởi 2 con trỏ. Các giá trị con trỏ được lưu trữ trong vùng dữ liệu của BIOS ở địa chỉ 40h:1Ah và 40h:1Ch. Con trỏ ghi (40h:1Ch) cho biết vị trí còn trống kế tiếp để ghi ký tự nhập, con trỏ đọc (40h:1Ah) cho biết vị trí ký tự đầu tiên sẽ đọc. Từ đó, bộ đệm bàn phím rỗng khi con trỏ ghi và con trỏ đọc trùng nhau → bộ đệm chỉ chứa được 15 ký tự.

Các hàm của ngắt 16h:

Hàm 0h - đọc ký tự từ bàn phím, nếu không nhấn thì sẽ chờ

Ra: AH = scancode, AL = mã ASCII. Nếu phím nhấn là các phím đặc biệt thì AL = 0

Hàm 1h - ZF = 1 nếu không có ký tự trong bộ đệm. Giá trị trả về giống như hàm 00h nhưng không xoá ký tự ra khỏi bộ đệm

Hàm 2h - Trả về trạng thái của các phím, kết quả chứa trong AL

7	6	5	4	3	2	1	0
INS	CAPS LOCK	NUM LOCK	SCROLL LOCK	ALT	CTRL	LEFT SHIFT	RIGHT SHIFT

Hàm 10h - Giống hàm 00h nhưng trả về mã mở rộng

Hàm 11h - Giống hàm 01h nhưng trả về mã mở rộng

Hàm 12h - Giống hàm 02h nhưng AH chứa thêm các thông tin

7	6	5	4	3	2	1	0
SYS REQ	CAPS LOCK	NUM LOCK	SCROLL LOCK	RIGHT ALT	RIGHT CTRL	LEFT ALT	LEFT CTRL

Các thí dụ:

- Giả sử phím 'c' đã được nhấn.

MOV AH,00h

INT 16h

Kết quả: AH = 2Eh (mã quét cho phím 'a'); AL = 63h (ASCII cho 'c')

- Giả sử phím 'HOME' đã được nhấn.

MOV AH,00h

INT 16h

Kết quả: AH = 47h (mã quét cho phím 'HOME')

AL = 0 (các phím chức năng và điều khiển không có mã ASCII)

- Giả sử phím 'HOME' đã được nhấn.

MOV AH,10h

INT 16h

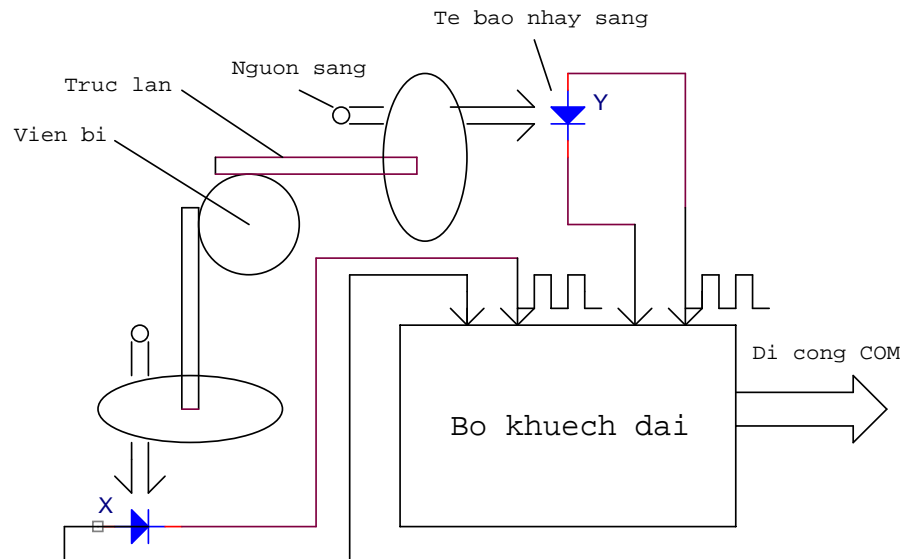
Kết quả: AH = 47h (mã quét cho phím 'HOME')

AL = E0h

2. Giao tiếp chuột

2.1. Cấu tạo

Cấu tạo của chuột rất đơn giản, phần trung tâm là 1 viên bi thép được phủ keo hoặc nhựa được quay khi dịch chuyển chuột. Chuyển động này được truyền tới 2 thanh nhỏ được đặt vuông góc với nhau. Các thanh này sẽ biến chuyển động của chuột theo 2 hướng X,Y thành sự quay tương ứng của 2 đĩa gắn với chúng. Trên 2 đĩa có những lỗ nhỏ liên tục đóng và ngắt 2 chùm sáng tới các sensor nhạy sáng để tạo ra các xung điện. Số các xung điện tỷ lệ với lượng chuyển động của chuột theo các hướng X,Y và số xung trên 1 sec biểu hiện tốc độ của chuyển động chuột. Kèm theo đó có 2 hay 3 phím bấm.



Hình 3.5 - Sơ đồ cấu tạo của chuột

2.2. Mạch ghép nối và chương trình điều khiển chuột

Hầu hết chuột được nối với PC qua cổng nối tiếp, qua đó chuột cũng được cấp nguồn nuôi từ PC. Khi dịch chuyển hoặc nhấn, nhả các phím chuột, nó sẽ phát ra một gói dữ liệu tới mạch giao tiếp và mạch sẽ phát ra 1 ngắt. Phần mềm điều khiển chuột làm các nhiệm vụ: chuyển ngắt tới mạch giao tiếp nối tiếp xác định, đọc dữ liệu và cập nhật các giá trị bên trong liên quan tới trạng thái của bàn phím cũng như vị trí của chuột. Hơn nữa, nó còn cung cấp 1 giao tiếp mềm qua ngắt 33h để định các giá trị bên trong này cũng như làm dịch chuyển con trỏ chuột trên màn hình tương ứng với vị trí của chuột.

Có thể chọn kiểu con trỏ chuột cứng hoặc mềm trong chế độ văn bản hay con trỏ chuột đồ họa trong chế độ đồ họa. Các hàm 09h và 0Ah trong ngắt 33h cho phép định nghĩa loại và dạng con trỏ chuột.

2.3. Chương trình với con trỏ

Ngắt 33h cho phép xác định vị trí, số lần click chuột và hình dạng con trỏ (số thứ tự hàm chứa trong AX).

Hàm	Ý nghĩa	Tham số
0	Reset chuột	Ra: AX = 0: nếu có, = 1: không BX = số nút nhấn
1	Hiển thị con trỏ	
2	Ẩn con trỏ	
3	Nhận vị trí con trỏ và trạng thái nút	Ra: BX: trạng thái nút (D0: nút trái, D1: nút phải, D2: nút giữa) (= 0: nhả, = 1: nhấn) CX: vị trí ngang DX: vị trí dọc
4	Đặt vị trí con trỏ	Vào: CX: vị trí ngang DX: vị trí dọc
5	Trạng thái nút và số lần nhấn từ khi gọi	Vào: BX = nút kiểm tra (=0: trái, =1: phải) Ra: AX = trạng thái nút BX = số lần nhấn CX: vị trí ngang DX: vị trí dọc lần nhấn cuối
6	Giống hàm 05h nhưng kiểm tra số lần nhả	
7	Giới hạn dịch chuyển ngang của con trỏ	Vào: CX = cột trái DX = cột phải
8	Giới hạn dịch chuyển dọc của con trỏ	Vào: CX = dòng dưới DX = dòng trên
9	Xác định hình dạng con trỏ đồ hoạ	Vào: BX = vị trí ngang CX = vị trí dọc ES:DX: địa chỉ mặt nạ màn hình và con trỏ
A	Xác định hình dạng con trỏ văn bản	Vào: BX = 0: con trỏ phần mềm CX = mặt nạ màn hình DX = mặt nạ con trỏ BX = 1: con trỏ phần cứng CX = dòng bắt đầu DX = dòng kết thúc

Chú ý rằng tọa độ con trỏ xác định theo pixel với độ phân giải 640x200 trong khi chế độ văn bản sử dụng tọa độ ký tự 80x25 nên để chuyển sang tọa độ ký tự thì phải chia cho 8. Con trỏ chuột hiển thị trên màn hình đồ hoạ bằng cách thực hiện:

Từ mới = (từ cũ AND mặt nạ màn hình) XOR mặt nạ con trỏ

Nếu ta đặt mặt nạ màn hình là 0 thì ký tự màn hình tại đó sẽ bị xoá.

VD: Con trỏ chuột mềm nhấp nháy và chứa ký tự 'A'

```
MOV AH, 0Ah
MOV BX, 0
MOV CX, 0 ; m•t n• màn hình = 0
MOV DH, 8Bh; = 10001011b → màu n•n Gray, màu ký t• Cyan
MOV DL, 'A'
INT 33h
```

VD: Con trỏ chuột cứng có các đường quét 3 và 8

```
MOV AH, 0Ah
MOV BX, 1
MOV CX, 03h
MOV DX, 08h
INT 33h
```

3. Giao tiếp màn hình

3.1. Card màn hình

Để hiện các hình ảnh, ký tự, hay hình vẽ trên màn hình, PC phải thông qua mạch ghép nối màn hình (graphics adapter). Board mạch này thường được cắm trên khe cắm mở rộng của PC.

Trong chế độ văn bản (text mode), các ký tự được xác định bởi mã ASCII, trong đó có cả các thông tin về thuộc tính của ký tự, thí dụ ký tự được hiện theo cách nhấp nháy hay đảo màu đen trắngROM ký tự (character rom) lưu trữ các hình mẫu điểm ảnh của các ký tự tương ứng để máy phát ký tự biến đổi các mã ký tự đó thành 1 chuỗi các bit điểm ảnh (pixel bit) và chuyển chúng tới thanh ghi dịch (shift register). Máy phát tín hiệu sẽ sử dụng các bit điểm ảnh này cùng với các thông tin thuộc tính từ RAM video và các tín hiệu đồng bộ từ CRTC để phát ra các tín hiệu cần thiết cho monitor.

Trong chế độ đồ họa (graphics mode), thông tin trong RAM video được sử dụng trực tiếp cho việc phát ra các ký tự. Lúc này các thông tin về thuộc tính cũng không cần nữa. Chỉ từ các giá trị bit trong thanh ghi dịch, máy phát tín hiệu sẽ phát các tín hiệu về độ sáng và màu cho monitor.

Mỗi ký tự được biểu diễn bởi 1 từ 2 byte trong RAM video. Byte thấp chứa mã ký tự, byte cao chứa thuộc tính. Cấu trúc của một từ nhớ video như sau:

15	14	13	12	11	10	9	8
BLNK	BAK ₂	BAK ₁	BAK ₀	INT	FOR ₂	FOR ₁	FOR ₀

7	6	5	4	3	2	1	0
CHR ₇	CHR ₆	CHR ₅	CHR ₄	CHR ₃	CHR ₂	CHR ₁	CHR ₀

BLNK: Nhấp nháy; 1 = bật, 0 = tắt
 INT: Cường độ sáng ; 1 = cao, 0 = bình thường
 CHR₇...CHR₀: Mã ký tự.

Bảng màu quy định như sau:

Mã hex	Màu	Mã hex	Màu	Mã hex	Màu	Mã hex	Màu
0	Black	4	Red	8	Dark Gray	C	Light Red
1	Blue	5	Magenta	9	Light Blue	D	Light Magenta
2	Green	6	Brown	A	Light Green	E	Yellow
3	Cyan	7	Light Gray	B	Light Cyan	F	White

Trong chế độ văn bản, 6845 liên tục xuất các địa chỉ cho RAM video qua MA0-MA13. Ký tự ở góc tận cùng phía trên bên trái màn hình có địa chỉ thấp nhất mà 6845 sẽ cung cấp ngay sau khi quét dọc ngược. Logic ghép nối định địa chỉ cho RAM video bằng việc lấy ra mã ký tự cùng với thuộc tính. Mã ký tự dùng cho máy phát ký tự như là chỉ số thứ nhất trong ROM ký tự. Lúc này, 6845 định địa chỉ hàng quét đầu tiên của ma trận ký tự, địa chỉ hàng bằng 0. Các bit của ma trận điểm ảnh bây giờ sẽ được truyền đồng bộ với tần số video từ thanh ghi dịch tới máy phát tín hiệu. Nếu máy phát tín hiệu nhận được giá trị 1 từ thanh ghi dịch, nó sẽ phát tín hiệu video tương ứng với màu của ký tự. Nếu nhận được 0 nó sẽ cấp tín hiệu tương ứng với màu nền. Vậy dòng quét thứ nhất được hiện phù hợp với các ma trận điểm ảnh của các ký tự trong hàng ký tự thứ nhất. Khi tia điện tử đạt tới cuối dòng quét, 6845 kích hoạt lối ra HS (Horizontal Synchronization) để tạo ra quá trình quét ngược và đồng bộ ngang. Tia điện tử quay trở về bắt đầu quét dòng tiếp. Sau mỗi dòng quét, 6845 tăng giá trị RA0-RA4 lên 1. Địa chỉ dòng này hình thành một giá trị offset bên trong ma trận điểm ảnh cho ký tự được hiện. Dựa trên mỗi dòng quét như vậy, một dòng các điểm ảnh của ký tự trong hàng ký tự được hiện ra. Điều này có nghĩa là với ma trận 9x14 điểm ảnh cho 1 ký tự, hàng ký tự thứ nhất đã được hiện sau 14 dòng quét. Khi địa chỉ RA0-RA4 trở về giá trị 0, 6845 sẽ cấp 1 địa chỉ MA0-MA13 mới và hàng ký tự thứ hai sẽ được hiện ra cũng như vậy. Ở cuối dòng quét cuối cùng, 6845 sẽ reset địa chỉ MA0-MA13 và RA0-RA4 và cho phép lối ra VS (Vertical Synchronization) phát ra tín hiệu quét ngược cùng tín hiệu đồng bộ dọc.

Mỗi ký tự có chiều cao cực đại ứng với 32 dòng vì có 5 đường địa chỉ RA0-RA4, còn bộ nhớ video trong trường hợp này được tới 16K từ vì có địa chỉ MA0-MA13 là 14 bit. Trong chế độ đồ họa, chúng kết hợp với nhau để tạo thành địa chỉ 19 bit, lúc đó 6845 có thể định địa chỉ cho bộ nhớ video lên tới 512K từ. Trong trường hợp này, các byte trong RAM video không được dịch thành mã ký tự và thuộc tính nữa mà trực tiếp xác định cường độ sáng và màu của điểm ảnh. Đa số các RAM video được chia thành vài băng được định địa chỉ bởi RA0-RA4. Các đường MA0-MA13 sẽ định địa chỉ offset bên trong mỗi băng. Dữ liệu trong RAM video lúc này được trực tiếp truyền tới thanh ghi dịch và máy phát tín hiệu. ROM ký tự và máy phát ký tự không làm việc.

RAM video được tổ chức khác nhau tùy theo chế độ hoạt động và bản mạch ghép nối. Thí dụ, với RAM video 128 KB, có thể địa chỉ hóa toàn bộ bộ nhớ màn hình qua CPU như bộ nhớ chính. Nhưng nếu kích thước RAM video lớn hơn thì làm như vậy sẽ đè lên vùng ROM mở rộng ở địa chỉ C0000h. Do đó, card EGA và VGA với trên 128 KB nhớ được tăng cường thêm 1 chuyển mạch mềm (soft-switch) cho phép thâm nhập các cửa sổ

128 KB khác nhau vào RAM video lớn hơn nhiều. Các chuyển mạch này được quy định bởi riêng các nhà sản xuất board mạch.

3.2. Chế độ văn bản

RAM video được coi như một dãy từ tuyến tính, từ đầu tiên được gán cho ký tự góc trên tận cùng bên trái màn hình gọi là hàng 1 cột 1. Từ thứ 2 là hàng 1, cột 2, Số từ tùy thuộc vào độ phân giải của kiểu hiện ký tự.

Thí dụ: độ phân giải chuẩn 25 hàng, 80 ký tự đòi hỏi 2000 từ nhớ 2 byte. Như vậy, tổng cộng cần 4 KB bộ nhớ RAM video. Trong khi đó với card có độ phân giải cao SVGA 60 hàng, 132 ký tự cần đến 15840 byte. Do đó RAM video thường được chia thành vài trang. Kích thước của mỗi trang tùy thuộc vào chế độ hiện của màn hình và số trang cực đại, phụ thuộc cả vào kích thước của RAM video. 6845 có thể được lập trình sao cho địa chỉ khởi phát của MA0-MA13 sau quét ngược dọc là khác 00h. Nếu địa chỉ khởi phát là bắt đầu của 1 trang thì có thể quản lý RAM video theo vài trang tách biệt nhau, nếu CPU thay đổi nội dung của 1 trang mà trang đó hiện đang không hiện thì màn hình cũng không thay đổi. Do đó, cần phân biệt trang nhớ đang được kích hoạt (đang hiện) và trang đang được xử lý.

Đoạn chương trình ghi ký tự 'A' có cường độ sáng cao vào góc trên bên trái với màu số 7 và màu nền số 0. Trang thứ nhất và là duy nhất bắt đầu ở địa chỉ B0000h.

```
MOV AX, 0B000h
MOV ES, AX
MOV AH, 0F8h
MOV AL, 41h
MOV ES:[00H], AX
```

3.3. Chế độ đồ họa

Tổ chức trong chế độ này phức tạp hơn. Ví dụ: với card Hercules, RAM video được chia thành 4 băng trên 1 trang. Băng thứ nhất: đảm bảo các điểm ảnh cho các dòng 0, 4, 8, ..., 344; băng thứ hai cho các dòng 1, 5, 9, ..., 345; băng thứ 3 cho các dòng 2, 6, 10, ..., 346; và băng thứ 4 cho các dòng 3, 7, 11, ..., 347. 64 KB được chia thành 2 trang 32 KB. Độ phân giải trong chế độ đồ họa là 720 x 348 điểm ảnh, mỗi điểm ảnh được biểu diễn bởi 1 bit. Do vậy, một dòng cần 90 byte (720 điểm ảnh / 8 điểm ảnh trên 1 byte). Địa chỉ của byte chứa điểm ảnh thuộc đường i và cột j trong trang k là:

$$B0000h + 8000h * k + 2000h * (i \bmod 4) + 90 * \text{int}(i/4) + \text{int}(j/8)$$

B0000h là đoạn video, 8000h là kích thước của trang, $2000h * (i \bmod 4)$ là offset của băng chứa byte đó, $90 * \text{int}(i/4)$ là offset của dòng i trong băng và $\text{int}(j/8)$ là offset của cột j trong băng.

3.4. Truy xuất màn hình qua DOS và BIOS

3.4.1. Truy xuất qua DOS

Các hàm của int 21h có thể hiện các ký tự trên màn hình nhưng không can thiệp được vào màu:

- Hàm 02h: ra màn hình.
- Hàm 09h: ra một chuỗi.
- Hàm 40h: ghi file/ thiết bị

Các lệnh **copy**, **type** và **print** trong command.com cho phép hiện text trên màn hình. DOS gộp chung bàn phím và monitor thành 1 thiết bị mang tên CON (console). Ghi CON là truyền số liệu tới monitor, còn đọc CON là nhận ký tự từ bàn phím. Ví dụ: để hiện nội dung của file output.txt lên màn hình của monitor sẽ có các cách sau:

- copy output.txt con
- type output.txt > con
- print output.txt /D:con

3.4.2. Truy xuất qua BIOS

BIOS thâm nhập monitor bằng int 10h với nhiều chức năng hơn DOS, như đặt chế độ hiện hình, quản lý tự động các trang, phân biệt các điểm trên màn hình nhờ các tọa độ,...

- **Hàm 00h**: định chế độ đồ họa:







Vào:

AL = chế độ

Mode	Type	Max Colors	Size	Resolution	Max Pages	Base Addr
----	-----	-----	----	-----	-----	-----
00	Text	16	40x25	- -	8	B8000h
01	Text	16	40x25	- -	8	B8000h
02	Text	16	80x25	- -	4,8	B8000h
03	Text	16	80x25	- -	4,8	B8000h
04	Graphics	4	40x25	320x200	1	B8000h
05	Graphics	4	40x25	320x200	1	B8000h
06	Graphics	2	80x25	640x200	1	B8000h
07	Text	Mono	80x25	- -	1,8	B0000h
08	Graphics	16	20x25	? ?	1	B0000h
09	Graphics	16	40x25	? ?	1	B0000h
0A	Graphics	4	80x25	? ?	1	B0000h
0B	- -	-	- -	- -	-	- -
0C	- -	-	- -	- -	-	- -
0D	Graphics	16	40x25	320x200	8	A0000h
0E	Graphics	16	80x25	640x200	4	A0000h
0F	Graphics	Mono	80x25	640x350	2	A0000h
10	Graphics	16	80x25	640x350	2	A0000h
11	Graphics	2	80x25	640x480	1	A0000h
12	Graphics	16	80x25	640x480	1	A0000h
13	Graphics	256	40x25	320x200	1	A0000h

- **Hàm 01h**: xác định hình dạng con trỏ

Vào:

0			
1			
2			
7			
	Đầu = 0 Cuối = 7	Đầu = 0 Cuối = 1	Đầu = 6 Cuối = 7

CH = dòng đầu con trỏ

CL = dòng cuối con trỏ

- **Hàm 02h**: xác định vị trí con trỏ

Vào:

DH = hàng, DL = cột (bắt đầu từ tọa độ 0,0)

BH = trang (= 0 trong chế độ đồ họa)

- **Hàm 03h**: lấy vị trí và hình dạng con trỏ

Vào: BH = trang

Ra:

DH = hàng, DL = cột

CH, CL = tham số xác định hình dạng con trỏ

- **Hàm 05h**: chọn trang màn hình

AL = số trang (0..7)

- **Hàm 06h**: cuộn cửa sổ lên

Vào:

AL = số hàng cuộn (= 0: cuộn toàn màn hình)

BH = thuộc tính các hàng trống xuất hiện khi cuộn

CH, CL = hàng, cột của góc phía trên bên trái

DH, DL = hàng, cột của góc phía dưới bên phải

- **Hàm 07h**: cuộn cửa sổ xuống

Giống hàm 06h

- **Hàm 08h**: đọc ký tự và thuộc tính ký tự tại vị trí con trỏ

Vào: BH = trang

Ra:

AH = thuộc tính

AL = mã ASCII

- **Hàm 09h**: ghi ký tự và thuộc tính ký tự tại vị trí con trỏ

Vào:

BH = trang

BL = thuộc tính (theo bảng trang 61)

AL = mã ASCII

CX = số ký tự cần xuất

- **Hàm 0Ah**: ghi ký tự tại vị trí con trỏ

Vào:

BH = trang

AL = mã ASCII

CX = số ký tự cần xuất

- **Hàm 0Bh**: chọn bảng màu (dùng cho chế độ đồ họa)

Vào: BH = 0, BL = chọn màu nền (theo bảng trang 61)

BH = 1, BL = số bảng màu

Palette	Pixel	Color	Palette	Pixel	Color
0	0	Black	1	0	Black
0	1	Green	1	1	Cyan
0	2	Red	1	2	Magenta
0	3	Brown	1	3	White

- **Hàm 0Ch**: hiện một điểm trên màn hình (dùng cho chế độ đồ họa)

AL = giá trị pixel (0 – 3), nếu AL.7 = 1 thì giá trị màu là phép toán XOR với giá trị màu hiện hành

CX = cột, DX = hàng

- **Hàm 0Dh**: đọc một điểm trên màn hình (dùng cho chế độ đồ họa)

Vào: CX = cột, DX = hàng

Ra: AL = giá trị pixel

- **Hàm 0Fh**: xác định chế độ màn hình hiện hành

Ra:

AL = chế độ

AH = số cột

BH = số trang

- **Hàm 13h**: xuất chuỗi

Vào:

AL = chế độ xuất (bit0 = 1: cập nhật vị trí con trỏ sau khi xuất, bit1 = 1: chuỗi có chứa thuộc tính ký tự)

BH = trang

BL = thuộc tính (theo bảng trang 61)

CX = số ký tự cần xuất

CX = số ký tự cần xuất

DH, DL = hàng, cột xuất chuỗi

ES:BP = chuỗi in (nếu AL.1 = 1 thì chuỗi có dạng 'Ký tự', thuộc tính, 'Ký tự', thuộc tính, 'Ký tự', thuộc tính...)

❖ Những thường trình đồ họa:

BIOS trên main board có sẵn những hàm dùng cho thâm nhập MDA và CGA. BIOS của riêng EGA và VGA có những hàm mở rộng tương ứng trong khi vẫn giữ nguyên định dạng gọi.

Một trong những hàm quan trọng nhất của int 10h là hàm 00h dùng để đặt chế độ hiện hình. Để thay đổi chế độ hiện hình cần phải làm rất nhiều bước chương trình phức tạp để nạp các thanh ghi của chip 6845. Trong khi đó, hàm 00h làm cho ta tắt cả các công việc này.

Thí dụ: tạo kiểu 6 với độ phân giải 640*200 trên CGA.

```
Mov ah, 00h      ; hàm 00h
Mov al, 06h      ; chế độ 6
Int 10h          ; gọi ngắt
```

Các card EGA/VGA có riêng BIOS của chúng. Trong quá trình khởi động PC, nó sẽ chặn int 10h lại và chạy chương trình BIOS riêng. Thường trình cũ (của BIOS trên board mạch chính) được thay địa chỉ tới int 42h. Tất cả các lệnh gọi int 10h sẽ được BIOS của EGA/VGA thay địa chỉ tới int 42h nếu board mạch EGA/VGA đang chạy các kiểu hiện tương thích với MDA hay CGA. Có các kiểu hoạt động từ 0 đến 7.

BIOS của EGA/VGA dùng vùng 40:84h tới 40:88h để lưu số liệu BIOS và các thông số của EGA/VGA. Nó có các hàm mới với các hàm phụ sau:

- Hàm 10h: truy xuất các thanh ghi màu và bảng màu
- Hàm 11h: cài đặt các bảng định nghĩa ký tự mới
- Hàm 12h: đặt cấu hình hệ con video
- Hàm 1Bh: thông tin về trạng thái và chức năng của BIOS video (chỉ có ở VGA)
- Hàm 1Ch: trạng thái save/restore của video (chỉ có ở VGA)

Sau đây là chức năng của các hàm và thí dụ sử dụng chúng:

- Hàm 10h, hàm phụ 03h – xoá/đặt thuộc tính

Ví dụ: Xoá thuộc tính nhấp nháy:

```
Mov ah, 10h      ; dùng hàm 10h
Mov al, 03h      ; dùng hàm phụ 03h
Mov bl, 00h      ; xoá thuộc tính nhấp nháy
Int 10h          ; gọi ngắt
```

- Hàm 11h – ghép nối với máy phát ký tự

Ví dụ: Nạp bảng định nghĩa ký tự 8*14 không cần chương trình CRTC:

```
Mov ah, 11h      ; dùng hàm 11h
Mov al, 01h      ; nạp bảng ký tự từ Rom Bios vào Ram máy phát ký tự.
Mov bl, 03h      ; gán số 3 cho bảng
```

Int 10h ; gọi ngắt

- Hàm 12h, hàm phụ 20h – chọn thường trình in màn hình. Dùng hàm phụ này có thể thay thế thường trình chuẩn cho INT 05h bằng thường trình có thể dùng cho các độ phân giải mới của EGA/VGA.

Ví dụ: Cho phép thường trình mới in màn hình:

Mov ah, 12h ; dùng hàm 12h

Mov bl, 20h ; dùng hàm phụ 20h

Ấn PRINT hoặc SHIFT+PRINT để gọi thường trình in đã được lắp đặt.

❖ Truy xuất trực tiếp bộ nhớ video:

Để vẽ 1 điểm trên màn hình, BIOS phải làm nhiều thủ tục nhưng nếu muốn vẽ toàn bộ 1 cửa sổ hình hay lưu trữ thì phải truy xuất trực tiếp RAM video.

- Với board đơn sắc MDA trong kiểu hiện văn bản số 7, 4 KB RAM được tổ chức như 1 dãy (array) gồm 2000 từ nhớ kề nhau (mỗi từ là mã thuộc tính: ký tự) tạo nên 25 dòng, 80 cột. RAM video bắt đầu ở đoạn B0000h, trong đó ký tự góc trên cùng bên trái là từ thứ nhất trong RAM video. Như vậy mỗi dòng có 160 byte (A0h). Địa chỉ của từ nhớ ứng với ký tự ở dòng i, cột j ($i = 0-24, j = 0-79$) được tính theo công thức sau:

$$\text{Address}(i,j) = B0000h + A0h * i + 02h * j.$$

- Với board EGA, ở kiểu hiện văn bản từ 0 đến 3 mã ký tự được lưu trữ trong lớp nhớ 0 cùng với thuộc tính trong lớp 1 của RAM video. Mạch logic chuyển địa chỉ trên board thực hiện sự kết hợp nhất định nào đó sao cho tổ chức và cấu trúc của RAM video cũng như cách tính địa chỉ vẫn tương đồng với cách của CPU. Trong chế độ đồ họa từ 13 đến 16, RAM video bắt đầu từ địa chỉ đoạn A000h. Các điểm ảnh được xếp kề cận nhau trong bộ nhớ và mỗi điểm ảnh đòi hỏi 4 bit, các bit này được phân ra ở 4 lớp nhớ. Như vậy địa chỉ của 1 trong 4 bit này trên 1 điểm ảnh không chỉ gồm đoạn video và offset mà còn thêm vào số lớp nhớ nữa.

Để hiện 1 điểm ảnh với 1 trong 16 màu, không phải chỉ tính địa chỉ bit mà còn phải thêm nhập 4 lớp nhớ. Muốn vậy, phải dùng thanh ghi mặt nạ bản đồ (map mask register). Thanh ghi này được định địa chỉ qua cổng chỉ số 3C4h với địa chỉ 02h và có thể được ghi qua cổng số liệu 3C5h. Cấu trúc của thanh ghi mặt nạ bản đồ như sau:

7	6	5	4	3	2	1	0
Res	Res	Res	Res	LY ₃	LY ₂	LY ₁	LY ₀

LY₃-LY₀: Thanh nhập ghi tới các lớp từ 0→3;

1 = cho phép; 0 = không cho phép

Res : Dự trữ

Ví dụ: Đặt bit 0 của byte ở địa chỉ A000:0000h cho lớp 0, 1, 3.

```
Mov AX, 0A000h    ; nạp đoạn video vào AX
Mov ES, AX         ; truyền đoạn video vào ES
Mov BX, 0000h      ; nạp offset 0000h vào BX
Out 3C4h, 02h      ; chỉ số 2 → thanh ghi mặt nạ bản đồ
Out 3C5h, 0Bh      ; ghi 0000 1011b vào thanh ghi mặt nạ bản đồ
                   ; (cho phép lớp 0, 1, 3)
Mov 3C5h, 0Bh      ; đặt bit 0 trong các lớp 0, 1 và 3
```

Để lưu trữ nội dung màn hình cần phải đọc các giá trị bit của 4 lớp khi dùng thanh ghi chọn bản đồ đọc (read map select register). Nó được định địa chỉ với chỉ số 04h qua cổng chỉ số 3CEh, và có thể được ghi qua cổng số liệu 3CFh. Cấu trúc của thanh ghi này:

7	6	5	4	3	2	1	0
res	res	res	res	res	res	LY ₁	LY ₀

LY₁-LY₀: cho phép thâm nhập đọc với:

00 = lớp 0

01 = lớp 1

10 = lớp 2

11 = lớp 3

res : dự trữ

Ví dụ: đọc byte ở địa chỉ A000:0000h cho lớp 2:

```
Mov AX, A000h      ; nạp đoạn video vào AX
Mov ES, AX         ; truyền đoạn video vào ES
Mov BX, 0000h      ; nạp offset vào BX
Out 3Ceh, 04h      ; chỉ số 4 → thanh ghi chọn bản đồ đọc
Out 3CFh, 02h      ; ghi 0000 0010b vào thanh ghi chọn bản đồ đọc
                   ; (cho phép lớp 2)
Mov AL, [ES:BX]    ; nạp byte trong lớp 2 vào AL.
```

Chú ý rằng 4 bit tại 4 lớp đại diện cho 1 điểm ảnh nên trong kiểu hiện 16 EGA có độ phân giải cao nhất mỗi dòng cần 80byte (640 điểm ảnh / 8 điểm ảnh trên 1 byte); mỗi trang màn hình gồm 32 KB. Địa chỉ byte của điểm ảnh ở dòng i, cột j trang k (i=0-349, j=0-639, k=0-1) là:

$$\text{Address}(i,j,k) = A0000h + 8000h * k + 50h * j + \text{int}(i/8).$$

Với board VGA, các chế độ hiện văn bản từ 0 đến 3 và 7 cũng như các chế độ đồ họa từ 4 đến 6 và 13 đến 16 của CGA. EGA và MDA đều chạy được trên nó.

Trong chế độ văn bản, mã ký tự được lưu trữ trong lớp nhớ 0 cùng với thuộc tính trong lớp 1 của RAM video VGA. Quá trình chuyển hóa địa chỉ cũng giống như EGA nhưng khác ở chỗ nó vẫn đảm bảo chế độ văn bản 7 với độ phân giải 720x400, ma trận điểm ảnh 9x16. Trong chế độ đồ họa 4÷6 và 13÷19, mọi tổ chức, cấu trúc cũng như cách tính

địa chỉ tương tự như CGA và EGA. VGA được tăng cường 3 kiểu hiện hình mới từ 17 đến 19.

Kiểu 17 tương thích với board đồ họa của máy PS/2 kiểu 30 là MCGA (multi colour graphics array). Các điểm ảnh chỉ gồm 1 bit (2 màu) được định vị chỉ trên lớp 0. Thí dụ, trong VGA kiểu 17 với 80 byte trên 1 dòng (640 điểm ảnh / 8 điểm ảnh trên 1 byte). Mỗi trang màn hình gồm 40 KB. Địa chỉ của byte ở dòng i, cột j (i= 0-479, j=0-639) như sau:

$$\text{Address}(i,j) = A0000h + 50h*j + \text{int}(i/8)$$

Kiểu 18, 4 bit của điểm ảnh được phân trong 4 lớp nhớ như ở EGA. Trong kiểu VGA phân giải cao với 16 màu khác nhau, 80 byte trên 1 dòng (640 điểm ảnh / 8 điểm ảnh trên 1 byte), mỗi trang màn hình gồm 40 KB (A0000h byte); địa chỉ của mỗi byte ở dòng i, cột j (i=0-479; j = 0-639) bằng:

$$\text{Address}(i,j) = A0000h + 50h*j + \text{int}(i/8).$$

Kiểu 19 với 256 màu cho 1 điểm ảnh thì RAM video lại được tổ chức rất đơn giản như 1 dãy tuyến tính, trong đó 1 byte tương ứng với 1 điểm ảnh. Giá trị của byte phân định màu của điểm ảnh. Kiểu này đòi hỏi 320 byte (140h) trên 1 dòng (320 điểm ảnh / 1 điểm ảnh trên 1 byte). Một trang màn hình gồm 64 KB (10000h) nhưng chỉ có 64000 byte được sử dụng. Địa chỉ của điểm ảnh trong dòng i, cột j (i = 0-199, j=0-319) là:

$$\text{Address}(i,j) = A0000h + 140h*j + i$$

PHỤ LỤC CHƯƠNG 3

```

TITLE    DISPLAYING MOUSE POSITION
CURSOR    MACRO ROW,COLUMN
            MOV    AH,02H
            MOV    BH,00
            MOV    DH,ROW
            MOV    DL,COLUMN
            INT     10H
            ENDM

DISPLAY  MACRO STRING
            MOV    AH,09H
            MOV    DX,OFFSET STRING
            INT     21H
            ENDM

.MODEL SMALL
.STACK
.DATA
MESSAGE_1    DB 'PRESS ANY KEY$'
MESSAGE_2    DB 'THE MOUSE CURSOR IS LOCATED AT $'
POS_HO       DB ?,?, ' AND $'
POS_VE       DB ?,?,'$'
OLDVIDEO     DB ?                ;current video mode
NEWVIDEO     DB 0EH              ;new video mode
.CODE
MAIN    PROC
            MOV    AX,@DATA
            MOV    DS,AX
            MOV    AH,0FH                ;get current video mode
            INT     10H
            MOV    OLDVIDEO,AL          ;save it
            MOV    AX,0600H              ;clear screen
            MOV    BH,07
            MOV    CX,0
            MOV    DX,184FH
            INT     10H
            MOV    AH,00H                ;set new video mode
            MOV    AL,NEWVIDEO
            INT     10H
            MOV    AX,0                  ;initialize mouse
            INT     33H
            MOV    AX,01                  ;show mouse cursor
            INT     33H
            CURSOR 20,20
            DISPLAY MESSAGE_1
AGAIN:
            MOV    AX,03H                ;get mouse location
            INT     33H
            MOV    AX,CX                ;get the hor. pixel position
            CALL    CONVERT
            MOV    POS_HO,AL            ;save the LSD
            MOV    POS_HO+1,AH          ;save the MSD

```

```

        MOV  AX,DX    ;get the vert. pixel position
        CALL CONVERT
        MOV  POS_VE,AL    ;save
        MOV  POS_VE+1,AH
        CURSOR 5,20
        DISPLAY MESSAGE_2
        DISPLAY POS_HO
        DISPLAY POS_VE
        MOV  AH,01          ;check for key press
        INT  16H
        JZ   AGAIN ;if no key press
        MOV  AH,02          ;hide mouse
        INT  33H
        MOV  AH,0          ;restore original video mode
        MOV  AL,OLDVIDEO    ;load original video mode
        MOV  AH,0 ;restore original video mode
        INT  10H
        MOV  AH,4CH          ;go back to DOS
        INT  21H

MAIN     ENDP
;-----
;divide pixels position by 8 and convert to ASCII to make
it displayable
;ax=pixels position (it is in hex)
;on return ax= two ASCII digits
CONVERT PROC
        SHR  AX,1          ;divide
        SHR  AX,1          ;by 8
        SHR  AX,1          ;to get screen position by character
        MOV  BL,10
        MOV  AH,0
        DIV  BL
        ADD  AX,3030H      ;make it ASCII
        RET              ;return with AX=two ASCII digits
CONVERT ENDP
END      MAIN
-----

;THIS PROGRAM WAITS FOR THE MOUSE PRESS COUNT AND
;DISPLAYS IT WHEN ANY KEY IS PRESSED.
.MODEL   SMALL
.STACK
.DATA
MESSAGE_1 DB 'PRESS LEFT BUTTON A NUMBER OF TIMES:LESS
THAN 99.$'
MESSAGE_2 DB 'TO FIND OUT HOW MANY TIMES, PRESS ANY
KEY$'
MESSAGE_3 DB 'YOU PRESSED IT $'
P_COUNT   DB '?,?', ' TIMES $'
MESSAGE_4 DB 'NOW PRESS ANY KEY TO GO BACK TO DOS$'
OLDVIDEO   DB ?
NEWVIDEO   DB 12H

```

```

.CODE
MAIN    PROC
    MOV     AX,@DATA
    MOV     DS,AX
    MOV     AH,0FH    ;get current video mode
    INT     10H
    MOV     OLDVIDEO,AL    ;save it
    MOV     AX,0600H    ;clear screen
    MOV     BH,07
    MOV     CX,0
    MOV     DX,184FH
    INT     10H
    MOV     AH,00H    ;set new video mode
    MOV     AL,NEWVIDEO
    INT     10H
    MOV     AX,0    ;initialize mouse
    INT     33H
    MOV     AX,01    ;show mouse cursor
    INT     33H
    CURSOR  2,1
    DISPLAY MESSAGE_1
    CURSOR  4,1
    DISPLAY MESSAGE_2
    MOV     AH,07    ;wait for key press
    INT     21H
    MOV     AX,05H    ;get mouse press count
    MOV     BX,0    ;check press count for left button
    INT     33H
    MOV     AX,BX    ;BX=button press count
    MOV     BL,10
    DIV     BL
    ADD     AX,3030H
    MOV     P_COUNT,AL    ;save the number
    MOV     P_COUNT+1,AH
    CURSOR  10,2
    DISPLAY MESSAGE_3
    DISPLAY P_COUNT
    CURSOR  20,2
    DISPLAY MESSAGE_4
    MOV     AH,07    ;wait for a key press to get out
    INT     21H
    MOV     AH,02    ;hide mouse
    INT     33H
    MOV     AH,0    ;restore original video mode
    MOV     AL,OLDVIDEO    ;load original vide mode
    INT     10H
    MOV     AH,4CH    ;go back to DOS
    INT     21H
MAIN    ENDP
    END MAIN
;-----

.MODEL  SMALL

```

```

.STACK 100h
.DATA
mask_mon DB 0FFh,0FFh,0FFh,0FFh,0FFh,0FFh,0FFh,0FFh
          DB 0FFh,0FFh,0FFh,0FFh,0FFh,0FFh,0FFh,0FFh
          DB 0FFh,0FFh,0FFh,0FFh,0FFh,0FFh,0FFh,0FFh
          DB 0FFh,0FFh,0FFh,0FFh,0FFh,0FFh,0FFh,0FFh
mask_p    DB 80h,0,0E0h,0,0F8h,0,0FEh,0
          DB 0D8h,0,0Ch,0,6,0,3,0
          DB 0,0,0,0,0,0,0,0
          DB 0,0,0,0,0,0,0,0
.CODE
main PROC
    mov ax,@data
    mov ds,ax
    mov es,ax

    mov ah,0
    mov al,6
    int 10h

    mov ax,0
    int 33h

    mov ax,1
    int 33h

    mov ah,08h
    int 21h

    mov ax,9
    mov bx,0
    mov cx,0
    lea dx,mask_mon
    int 33h

    mov ah,08h
    int 21h

    mov ah,4Ch
    int 21h
main ENDP
END main

```

```

1000000000000000
1110000000000000
1111100000000000
1111111000000000
1101100000000000
0000110000000000
0000011000000000
0000001100000000

```

```

0000000000000000
0000000000000000
0000000000000000
0000000000000000
0000000000000000
0000000000000000
0000000000000000
0000000000000000

```

Keyboard Scan Codes: Set 1

*All values are in hexadecimal

101-, 102-, and 104-key keyboards:

KEY	MAKE	BREAK	-----	KEY	MAKE	BREAK	-----	KEY	MAKE	BREAK
A	1E	9E		9	0A	8A		[1A	9A
B	30	B0		`	29	89		INSERT	E0,52	E0,D2
C	2E	AE		-	0C	8C		HOME	E0,47	E0,97
D	20	A0		=	0D	8D		PG UP	E0,49	E0,C9
E	12	92		\	2B	AB		DELETE	E0,53	E0,D3
F	21	A1		BKSP	0E	8E		END	E0,4F	E0,CF
G	22	A2		SPACE	39	B9		PG DN	E0,51	E0,D1
H	23	A3		TAB	0F	8F		U ARROW	E0,48	E0,C8
I	17	97		CAPS	3A	BA		L ARROW	E0,4B	E0,CB
J	24	A4		L SHFT	2A	AA		D ARROW	E0,50	E0,D0
K	25	A5		L CTRL	1D	9D		R ARROW	E0,4D	E0,CD
L	26	A6		L GUI	E0,5B	E0,DB		NUM	45	C5
M	32	B2		L ALT	38	B8		KP /	E0,35	E0,B5
N	31	B1		R SHFT	36	B6		KP *	37	B7
O	18	98		R CTRL	E0,1D	E0,9D		KP -	4A	CA
P	19	99		R GUI	E0,5C	E0,DC		KP +	4E	CE
Q	10	19		R ALT	E0,38	E0,B8		KP EN	E0,1C	E0,9C
R	13	93		APPS	E0,5D	E0,DD		KP .	53	D3
S	1F	9F		ENTER	1C	9C		KP 0	52	D2
T	14	94		ESC	01	81		KP 1	4F	CF
U	16	96		F1	3B	BB		KP 2	50	D0
V	2F	AF		F2	3C	BC		KP 3	51	D1
W	11	91		F3	3D	BD		KP 4	4B	CB
X	2D	AD		F4	3E	BE		KP 5	4C	CC
Y	15	95		F5	3F	BF		KP 6	4D	CD
Z	2C	AC		F6	40	C0		KP 7	47	C7

0	0B	8B		F7	41	C1		KP 8	48	C8
1	02	82		F8	42	C2		KP 9	49	C9
2	03	83		F9	43	C3]	1B	9B
3	04	84		F10	44	C4		;	27	A7
4	05	85		F11	57	D7		'	28	A8
5	06	86		F12	58	D8		,	33	B3
6	07	87		PRNT SCRN	E0,2A, E0,37	E0,B7, E0,AA		.	34	B4
7	08	88		SCROLL	46	C6		/	35	B5
8	09	89		PAUSE	E1,1D,45 E1,9D,C5	-NONE-				

Keyboard Scan Codes: Set 2

*All values are in hexadecimal

101-, 102-, and 104-key keyboards:

KEY	MAKE	BREAK	-----	KEY	MAKE	BREAK	-----	KEY	MAKE	BREAK
A	1C	F0,1C		9	46	F0,46		[54	F0,54
B	32	F0,32		`	0E	F0,0E		INSERT	E0,70	E0,F0,70
C	21	F0,21		-	4E	F0,4E		HOME	E0,6C	E0,F0,6C
D	23	F0,23		=	55	F0,55		PG UP	E0,7D	E0,F0,7D
E	24	F0,24		\	5D	F0,5D		DELETE	E0,71	E0,F0,71
F	2B	F0,2B		BKSP	66	F0,66		END	E0,69	E0,F0,69
G	34	F0,34		SPACE	29	F0,29		PG DN	E0,7A	E0,F0,7A
H	33	F0,33		TAB	0D	F0,0D		U ARROW	E0,75	E0,F0,75
I	43	F0,43		CAPS	58	F0,58		L ARROW	E0,6B	E0,F0,6B
J	3B	F0,3B		L SHFT	12	F0,12		D ARROW	E0,72	E0,F0,72
K	42	F0,42		L CTRL	14	F0,14		R ARROW	E0,74	E0,F0,74
L	4B	F0,4B		L GUI	E0,1F	E0,F0,1F		NUM	77	F0,77
M	3A	F0,3A		L ALT	11	F0,11		KP /	E0,4A	E0,F0,4A
N	31	F0,31		R SHFT	59	F0,59		KP *	7C	F0,7C
O	44	F0,44		R CTRL	E0,14	E0,F0,14		KP -	7B	F0,7B
P	4D	F0,4D		R GUI	E0,27	E0,F0,27		KP +	79	F0,79
Q	15	F0,15		R ALT	E0,11	E0,F0,11		KP EN	E0,5A	E0,F0,5A
R	2D	F0,2D		APPS	E0,2F	E0,F0,2F		KP .	71	F0,71
S	1B	F0,1B		ENTER	5A	F0,5A		KP 0	70	F0,70
T	2C	F0,2C		ESC	76	F0,76		KP 1	69	F0,69
U	3C	F0,3C		F1	05	F0,05		KP 2	72	F0,72
V	2A	F0,2A		F2	06	F0,06		KP 3	7A	F0,7A
W	1D	F0,1D		F3	04	F0,04		KP 4	6B	F0,6B

X	22	F0,22		F4	0C	F0,0C		KP 5	73	F0,73
Y	35	F0,35		F5	03	F0,03		KP 6	74	F0,74
Z	1A	F0,1A		F6	0B	F0,0B		KP 7	6C	F0,6C
0	45	F0,45		F7	83	F0,83		KP 8	75	F0,75
1	16	F0,16		F8	0A	F0,0A		KP 9	7D	F0,7D
2	1E	F0,1E		F9	01	F0,01]	5B	F0,5B
3	26	F0,26		F10	09	F0,09		;	4C	F0,4C
4	25	F0,25		F11	78	F0,78		'	52	F0,52
5	2E	F0,2E		F12	07	F0,07		,	41	F0,41
6	36	F0,36		PRNT SCRN	E0,12, E0,7C	E0,F0, 7C,E0, F0,12		.	49	F0,49
7	3D	F0,3D		SCROLL	7E	F0,7E		/	4A	F0,4A
8	3E	F0,3E		PAUSE	E1,14,77, E1,F0,14, F0,77	-NONE-				

AT Keyboard Scan Codes (Set 3)

KEY	MAKE	BREAK	-----	KEY	MAKE	BREAK	-----	KEY	MAKE	BREAK
A	1C	F0,1C		9	46	F0,46		[54	F0,54
B	32	F0,32		`	0E	F0,0E		INSERT	67	F0,67
C	21	F0,21		-	4E	F0,4E		HOME	6E	F0,6E
D	23	F0,23		=	55	F0,55		PG UP	6F	F0,6F
E	24	F0,24		\	5C	F0,5C		DELETE	64	F0,64
F	2B	F0,2B		BKSP	66	F0,66		END	65	F0,65
G	34	F0,34		SPACE	29	F0,29		PG DN	6D	F0,6D
H	33	F0,33		TAB	0D	F0,0D		U ARROW	63	F0,63
I	43	F0,48		CAPS	14	F0,14		L ARROW	61	F0,61
J	3B	F0,3B		L SHFT	12	F0,12		D ARROW	60	F0,60
K	42	F0,42		L CTRL	11	F0,11		R ARROW	6A	F0,6A
L	4B	F0,4B		L WIN	8B	F0,8B		NUM	76	F0,76
M	3A	F0,3A		L ALT	19	F0,19		KP /	4A	F0,4A
N	31	F0,31		R SHFT	59	F0,59		KP *	7E	F0,7E
O	44	F0,44		R CTRL	58	F0,58		KP -	4E	F0,4E
P	4D	F0,4D		R WIN	8C	F0,8C		KP +	7C	F0,7C
Q	15	F0,15		R ALT	39	F0,39		KP EN	79	F0,79
R	2D	F0,2D		APPS	8D	F0,8D		KP .	71	F0,71
S	1B	F0,1B		ENTER	5A	F0,5A		KP 0	70	F0,70
T	2C	F0,2C		ESC	08	F0,08		KP 1	69	F0,69

U	3C	F0,3C		F1	07	F0,07		KP 2	72	F0,72
V	2A	F0,2A		F2	0F	F0,0F		KP 3	7A	F0,7A
W	1D	F0,1D		F3	17	F0,17		KP 4	6B	F0,6B
X	22	F0,22		F4	1F	F0,1F		KP 5	73	F0,73
Y	35	F0,35		F5	27	F0,27		KP 6	74	F0,74
Z	1A	F0,1A		F6	2F	F0,2F		KP 7	6C	F0,6C
0	45	F0,45		F7	37	F0,37		KP 8	75	F0,75
1	16	F0,16		F8	3F	F0,3F		KP 9	7D	F0,7D
2	1E	F0,1E		F9	47	F0,47]	5B	F0,5B
3	26	F0,26		F10	4F	F0,4F		;	4C	F0,4C
4	25	F0,25		F11	56	F0,56		'	52	F0,52
5	2E	F0,2E		F12	5E	F0,5E		,	41	F0,41
6	36	F0,36		PRNT SCRN	57	F0,57		.	49	F0,49
	3D	F0,3D		SCROLL	5F	F0,5F		/	4A	F0,4A
8	3E	F0,3E		PAUSE	62	F0,62				