Ram Video

Per tutti !!

- La **Ram Video** è proprio uno dei segreti del tuo computer; la sua presenza è del tutto trasparente per tutte le applicazioni e i programmi, cioè ***c'è ma non si vede*** (come il trucco...).

- Eppure nessun applicativo può farne a meno; in pratica ogni volta che viene creata una schermata, sia con *contenuto grafico* (tipico dell'ambiente Windows) sia in "*modo testo*" (tipico dell'ambiente Dos) viene coinvolta la **Ram Video**.

- Il nome di questa struttura deriva dalla sua collocazione (Ram) e dal suo scopo (gestire il Video):
  - la memoria **Ram** è uno tra i più comuni e più importanti dispositivi del tuo computer; la sua presenza è vitale per il suo funzionamento; quando lo acquisti puoi scegliere se installarne 32M, o 64M, o 128M, o in misura maggiore, se il portamonete te lo consente... Il suo nome è l'acronimo di **R**andom **A**ccess **M**emory, *memoria ad accesso casuale*, e sembra poco adatto a descrivere il meccanismo con il quale viene effettivamente usata
  - il **Video** è un termine generico, improprio, con cui si suole chiamare il sottoinsieme di dispositivi chiamati a mostrare le informazioni prodotte da un computer; spesso viene confuso con il *Monitor* (una specie di televisore di grande qualità) ma con esso bisogna coinvolgere anche la *scheda video*; le due cose non possono vivere... separatamente.

- In generale la Ram è come un taccuino in cui si annotano le informazioni utili; la **Ram Video** viene scritta dal processore con dati che riguardano l'immagine o il testo da mostrare sul monitor.

- Il compito di trasferire sul **monitor** le informazioni spetta alla scheda video: essa le attinge da questa memoria e le sottopone ai suoi dispositivi interni; ciascun byte viene alla fine trasformato in una serie di segnali elettrici, tre per ogni puntino (*pixel*) dello schermo.

- Il segnale di ogni terna viene applicato ad uno dei tre "cannoni" contenuti dentro il monitor, più grande è la sua intensità maggiore è la forza dello sparo. Il "proiettile" di questi cannoni è un fascio di elettroni; il bersaglio, un piccolo fosforo colorato di blu (o di verde o di rosso, un colore per cannone).

- In conclusione l'informazione digitale scritta dal processore si traduce in un puntino nel colore ottenuto dalla fusione dei tre fondamentali, a loro volta accesi più o meno intensamente dai rispettivi cannoni.

- E' importante sottolineare che l'azione del *processore* non produce alcun effetto visivo; questo compito spetta alla *scheda video*.
- I due dispositivi hanno in comune la **Ram Video** cosicché mentre *la seconda* ne "spazzola" il contenuto *il primo* può cambiarle le carte in tavola…

- Per finire ricordiamo che **nel primo megaByte** trovano storicamente posto delle aree dal contenuto fondamentale: una dei queste *zone di memoria* è la **Ram Video**.

Come Funziona

PRESENTAZIONE

- La **Ram Video** è una zona della **memoria Convenzionale di Sistema**; la sua dimensione è di **128**kBytes, a partire dall'indirizzo fisico **A0000H** fino all'indirizzo fisico **BFFFFH** nel primo megaByte; la pratica della programmazione a basso livello consente di esprimere questi indirizzi fisici nei corrispondenti indirizzi logici, da *A000:0000* a *B800:7FFF* (un indirizzo logico si esprime sempre nella forma *Segmento*:*Offset*).

- Lo studio della **Ram Video** non può essere separato da quello delle ***Funzioni*** e ***Procedure*** dedicate al controllo delle numerose modalità di funzionamento di una *scheda video*, sia livello ***Bios*** che a livello ***Dos***.
- Di fatto, pur nella sua originale identità, essa è pur sempre la porta hardware attraverso la quale vengono generate le immagini poste a video.

La scrittura in **Ram Video** è il modo più veloce per porre *qualcosa* sul monitor; l'operazione si effettua ***con una sola istruzione***, senza coinvolgere nessun altro strato software (ne ***Bios*** ne ***Dos*** ne altro)

- Il rapporto di scrittura è con la *memoria*, non con un *dispositivo di I/O*; per questo vi si accede con un ***MOV*** e non con un ***OUT***; per questo è velocissimo e immediato.

La consapevolezza di aver a che fare con *memoria* può insinuare anche il desiderio di *leggere* la **Ram Video**: questa opportunità apre orizzonti impensabili, per esempio consentendo al programmatore di "trovare" oggetti nascosti alla vista semplicemente per il fatto di essere stati precedentemente stampati in nero su nero (ottimo per le battaglie navali o per i bonus di un gioco!).

- Questa area di memoria è a sua volta divisa in tre parti:
  - nella prima (da *A000:0000* a *A000:FFFF*) trovano posto i **64**K bytes destinati alla gestione del video in **Modo Grafico**. L'unità di informazione è il **pixel** (semplificando, ciascun byte (8 bit) di quest'area può rappresentare un solo punto grafico, o meglio uno dei 2^8=256 colori che esso può assumere; ma anche otto pixel, nel qual caso ciascuno di essi non può assumere solo due colori, bianco o nero, cioè acceso o spento).
  - nella seconda (da *B000:0000* a *B000:7FFF*) sono disponibili i **32**K bytes dedicati al **modo video Monocromatico**, ora in disuso.
  - nell'ultima parte (da *B000:8000* a *B000:FFFF*) sono allocati i **32**K bytes necessari alla gestione del video in **Modo Testo**. L'unità di informazione è il **carattere** (vedremo che per descrivere le caratteristiche di ciascuno di essi saranno necessari due byte di quest'area).

- La differenza tra "*Modo Grafico*" e "*Modo Testo*" sta dunque nel modo con cui vengono trattati i punti (cioè le terne di fosfori, blu verdi e rossi) del tuo monitor. Nel primo si interviene singolarmente su ciascun pixel; nel secondo vengono erogati (di norma) 128 pixel alla volta, tanti quanti sono necessari per "disegnare" sullo schermo un carattere, appunto creato a partire da una matrice di 8x16 punti (di questo si occupa il generatore di caratteri della scheda video).

- Le pagine seguenti consentono di approfondire la conoscenza delle ***tre aree*** **di RamVideo**.

Come Funziona

MODO GRAFICO

- Nei **modi grafici** l'area di **memoria Convenzionale** coinvolta dalla Scheda Video è di **64**kBytes, a partire dall'indirizzo fisico **A0000H** fino all'indirizzo fisico **AFFFFH** nel primo megaByte, corrispondente agli indirizzi logici da *A000:0000* a *B800:7FFF*.
- L'analisi e la programmazione della parte di Ram Video destinata ai modi grafici è dunque ragionevole solo per i più semplici (cioè quelli a *minor risoluzione*): solo per essi è possibile il controllo diretto con la semplice scrittura diretta (***MOV***). Vediamo di capire il perché.

- Con la risoluzione **320*200**, ormai retaggio storico ma ampiamente usata dai vecchi giochi Dos, sono necessari 64000 pixel (320*200=64000): la quantità di pixel è perciò inferiore al numero di bytes disponibili (ne avanzano 1536). In questo caso è possibile associare un byte della **Ram Video** ad ogni pixel.
- Poiché gli otto bit di ciascun byte possono essere combinati in 2^8 = 256 modi possiamo concludere che, con la risoluzione **320*200**, le immagini grafiche proposte a video hanno **profondità di colore di 8 bit**, cioè ciascun suo pixel *può assumere al massimo **256** diversi colori*.

- Il lettore attento ha già concluso che il meccanismo descritto comincia a scricchiolare non appena saliamo anche di poco con la risoluzione; già con una normalissima **800*600** i punti (pixel) necessari passano a 800*600 = 480000.
- E' dunque impossibile associare ciascun pixel ad un byte, come succedeva prima; al più possiamo tentare di associarlo a un bit; solo così tutti i 480000 punti richiesti dalla risoluzione potranno essere comodamente ospitati tra i 524288 bit (=65536*8) della Ram Video.
- La risoluzione **800*600** è così certamente garantita, ma l'immagine potrà essere solo **a 2 colori** (per esempio **in bianco e nero**): infatti ogni byte della **Ram Video** rappresenterà 8 pixel *con i suoi 8 bit* e, potendo un bit essere o ***0*** o ***1***, i pixel potranno essere o ***spenti*** o ***accesi***.

- Ma cosa succede se la risoluzione **800*600** viene utilizzata in modo normale, per esempio **a 65536 colori**? Per codificare ciascun colore servono **16 bit** (2^{16} = 65536 diverse possibilità) cioè 2 bytes; qualunque immagine posta a video ha dunque bisogno di 800*600*2 = 960000 bytes = 937.5 kBytes, **più di quindici volte** la memoria resa disponibile dalla **Ram Video** (=64k).
- Dunque, deve esserci qualche inghippo che assicuri la gestione dei 480 mila pixel.

I *65536 bytes* della **Ram Video** destinata alla Grafica sono usati in modo di volta in volta diverso, in funzione della **risoluzione** scelta per la stampa dei pixel dell'immagine.
Non è pensabile approfondire in questa sede il loro utilizzo per tutte le possibili risoluzioni: i meccanismi che regolano quest'area di memoria sono oggettivamente molto complessi e saranno trattati solo quando parleremo della programmazione a basso livello di una ***scheda grafica***.

- Senza entrare nel merito, possiamo concludere che, in questo caso, la **RamVideo** non è che la *parte emergente di un iceberg*; i suoi 65536*8 = 524288 bit rappresentano il primo (il bit0) degli **n** bit necessari a codificare il colore.
- Nel nostro esempio, con **16 bit di colore** saranno dunque necessarie altre quindici aree uguali alla **RamVideo Grafica**, supposte "sottostanti" ad essa, come in un pacco di fogli.
- Si tratta del concetto dei **piani di bit**, dal quale prende forma anche il concetto di **profondità** *di colore*: la profondità (lo spessore del pacco) è tanto maggiore quanti più bit sono necessari per descrivere il colore di ciascun pixel.
- Naturalmente la memoria necessaria per realizzare questo speciale wafer di bit è ospitata sulla ***scheda Video***: maggiore è la quantità di Ram presente sulla scheda maggiore è la risoluzione e la profondità di colore possibile.
- Per finire può essere utile un rimando alle ricche pagine che si occupano di descrivere i colori nei **Modi Grafici**.

Come Funziona — MODO MONOCROMATICO

- Nei **modi monocromatici** l'area di **memoria Convenzionale** coinvolta dalla ***scheda Video*** è di **32**kBytes, a partire dall'indirizzo fisico **B0000H** fino all'indirizzo fisico **B7FFFH** nel primo megaByte, corrispondente agli indirizzi logici da *B000:0000* a *B000:7FFF*.

- Poichè di *Monitor monocromatici* non si parla proprio più, i bytes di quest'area non sono più utilizzati da nessuno, sebbene sia ancora possibile gestirli con proprietà.

- Si tratta di 32768 bytes "liberi"; può essere interessante ricordarselo, quando si programma in Assembler....

Come Funziona — MODO TESTO

- Quando si lavora in **Modo Testo** la zona di **Ram Video** che ci interessa è quella compresa tra *B000:8000* e *B000:FFFF*; poiché la tecnica della *segmentazione della memoria* ci consente di esprimere il *medesimo indirizzo logico* in molti modi diversi, per motivi di opportunità conviene modificare il contenuto *Segment:Offset* di questo intervallo, trasformandolo in **B800:0000** - **B800:7FFF**.
- I **32** kBytes della **Ram Video** destinata al **Modo Testo** sono divisi in **otto** zone uguali, dette **Pagine Video**, ciascuna di **4** kBytes (4096 bytes).
- Ciascuna delle otto Pagine Video, numerate da **0** a **7**, copre quindi l'area di memoria Ram corrispondente agli indirizzi:

Numero Pagina	Primo Address	Range Indirizzi Fisici
Pagina 0	B800:0000	da B8000 a B8FFF
Pagina 1	B800:1000	da B9000 a B9FFF
Pagina 2	B800:2000	da BA000 a BAFFF
Pagina 3	B800:3000	da BB000 a BBFFF
Pagina 4	B800:4000	da BC000 a BCFFF
Pagina 5	B800:5000	da BD000 a BDFFF
Pagina 6	B800:0600	da BE000 a BEFFF
Pagina 7	B800:0700	da BF000 a BFFFF

- In **Modo Testo** la memoria **RamVideo** viene utilizzata *a livello di byte* (piuttosto che *a livello di bit*, tipico della gestione grafica); le informazioni necessarie per colorare i 128 pixel (8*16) punti necessari per "disegnare" sullo schermo un carattere, sono dunque affidate a due bytes.
- Lo sviluppo dei programmi (sia in *Turbo Pascal* che in *Assembler*) viene di norma fatto in ambiente DOS che, tipicamente, è una espressione video in Modo Testo.

La **Ram Video** che ci interessa è divisa in **otto parti**, dette **Pagine Video**. Puoi pensare alle **8 Pagine Video** come a una catasta di **cassetti**, ognuno dei quali contiene una sequenza di ***2000 oggetti rettangolari*** (i *caratteri*) organizzati su ***25 righe*** da ***80 colonne*** ciascuna. Naturalmente puoi vedere solo il contenuto del cassetto in cima alla pila (la **Pagina 0**) mentre quello degli altri (le Pagine da 1 a 7) c'è ma non si vede (*come il trucco...*). La magia consiste nel ***spostare*** *tutti o in parte* ***gli oggetti da un cassetto all'altro***, quando vuoi, ***non dimenticando*** *che comunque* ***potrai vedere solo il contenuto del cassetto in cima*** *alla catasta.*

- Il concetto principale sta dunque nel fatto che **solo** la **Pagina 0** (cioè i primi 4096 bytes di questa area) **viene letta continuamente** dalla scheda video e il suo contenuto viene interpretato e tradotto direttamente sul monitor. Ogni modifica (***scrittura***) eseguita sui primi 4000 bytes si traduce in una modifica in tempo reale sull'aspetto del testo mostrato sul monitor. Possiamo dunque concludere:
  - solo quello che viene ***scritto*** in **Pagina 0** produce effetto a video
  - le eventuali modifiche eseguite sul contenuto delle rimanenti sette Pagine (da **Pagina 1** a **Pagina 7**) ***non si vede!***, per cui la presenza di queste Pagine **sembra** inutile
  - in realtà la fortuna di disporre di queste sette pagine alternative ci offre la possibilità di "salvare" in esse tutta o in parte la **Pagina 0**; questa operazione risulta indispensabile per esempio quando si desidera scrivere un messaggio di avviso o di errore sull'immagine corrente, senza perdere il testo originale quando il messaggio viene tolto
  - è sufficiente infatti **salvare** l'area coperta dal messaggio in una *Pagina alternativa* **prima** di stampare il messaggio, per **poi recuperarla** in Pagina 0 non appena il messaggio è stato letto.

Quando lavori in *Turbo Pascal* il monitor ti mostra un'area (la **Pagina 0**) organizzata in **25 righe**, ciascuna di **80 caratteri**. Digitando sulla tastiera puoi quindi scrivere **fino a 2000** (=**25**x**80**) **caratteri**, per i quali puoi prevedere sia il colore di primo piano (cioè il colore del carattere stesso) sia quello dello sfondo (cioè la tinta del rettangolino che ospita ciascun carattere). Continuando con la metafora precedente per ogni carattere che digiti viene inserito un blocchetto del *cassetto in cima alla pila*.

- Rimane da chiarire in che modo i 4096 bytes di una pagina vengono usati per rappresentare i *2000 caratteri* da essa ospitati Bisogna sapere che ciascun carattere ha bisogno di *due bytes* (uno per il codice Ascii e uno per il codice di colore) per cui possiamo concludere che sono necessari **4000** bytes (dei 4096 disponibili).

- Ogni Pagina Video ha quindi a disposizione più bytes di quanti ne servano effettivamente
  - Il ***codice Ascii*** *del carattere* è il numero che lo rappresenta; consulta la Tabella dei Codici Ascii per saperne di più
  - Il ***codice di colore*** (detto anche ***attributo***) *del carattere* è un byte costruito dividendo gli otto bit in tre campi, secondo il seguente schema:

bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0
flash	Sfondo			Primo Piano			
F	S2	S1	S0	P3	P2	P1	P0

  il colore associato a ciascun campo è descritto nella pagina seguente; la combinazione degli 8 bit da luogo a 2^8 = 256 diversi accoppiamenti: puoi vederne l'effetto ed impararne il codice consultando la pagina dei Colori Assembler in Modo Testo

- Il campo più a sinistra è formato dal **solo bit7**: se esso vale "1" lo sfondo viene fatto lampeggiare, mentre con bit7="0" lo sfondo appare normalmente.

Il campo più a destra è costituito da **quattro bit** e consente quindi sedici combinazioni; a ciascuna di esse corrisponde un **colore di Primo Piano**, da 0000=*Nero* a 1111=*Bianco Brillante*, in accordo con la seguente Tabella:

bit3	bit2	bit1	bit0	Primo Piano		
0	0	0	0	Black	**0**	**Nero**
0	0	0	1	Blue	**1**	**Blu**
0	0	1	0	Green	**2**	**Verde**
0	0	1	1	Cyan	**3**	**Azzurro**
0	1	0	0	Red	**4**	**Rosso**
0	1	0	1	Magenta	**5**	**Magenta**
0	1	1	0	Brown	**6**	**Marrone**
0	1	1	1	LightGray	**7**	**Bianco**
1	0	0	0	darkGray	**8**	**Grigio**
1	0	0	1	LightBlue	**9**	**Blu Elettrico**
1	0	1	0	LightGreen	**10**	**Verde Chiaro**
1	0	1	1	LightCyan	**11**	**Celeste**
1	1	0	0	LightRed	**12**	**Rosa**
1	1	0	1	LightMagenta	**13**	**Magenta Chiaro**
1	1	1	0	Yellow	**14**	**Giallo**
1	1	1	1	White	**15**	**Bianco Brillante**

Il campo centrale è costituito da **tre bit** e consente quindi otto combinazioni; a ciascuna di esse corrisponde un **colore dello Sfondo**, da 000=*Nero* a 111=*Bianco*, in accordo con la seguente Tabella:

bit6	bit5	bit4	Sfondo		
0	0	0	Black	**0**	**Nero**
0	0	1	Blue	**1**	**Blu**
0	1	0	Green	**2**	**Verde**
0	1	1	Cyan	**3**	**Azzurro**
1	0	0	Red	**4**	**Rosso**
1	0	1	Magenta	**5**	**Magenta**
1	1	0	Brown	**6**	**Marrone**
1	1	1	LightGray	**7**	**Bianco**