128 da zero

di Andrea de Prisco

MMU: scottanti rivelazioni

MMU sta per Memory Management Unit. Nella sua accezione più classica, una MMU serve per tradurre un indirizzo logico in un indirizzo fisico. Molte volte, per motivi di praticità, sono demandati alle MMU altri compiti sempre inerenti la gestione della memoria. Con questo articolo vedremo cosa fa la MMU all'interno del 128, oltre naturalmente a gestire i banchi di memoria discussi alcuni numeri fa.

Cos'è una MMU

Prima di entrare nel merito *centovettotesimo* desideriamo dare alcuni chiarimenti circa l'accezione classica di MMU. Come descritto in Appunti di Informatica di MC numero 53, i calcolatori *veri* dispongono del meccanismo della memoria virtuale. Ciò al fine di ottimizzare l'utilizzo della memoria fisica, da parte dei vari processi in esecuzione.

Semplicisticamente parlando, per ogni programma in esecuzione non è mantenuto in memoria centrale tutto il codice e tutti i dati, ma solo un sottoinsieme di questi necessari per l'elaborazione in quel momento. Se un determinato dato o un pezzo di codice è richiesto, ma non è contenuto in memoria, il sistema provvede a prelevarlo dalla memoria secondaria (dischi) eventualmente scaricando qualcos'altro per «fare posto». A causa di questo fatto, lo spazio di indirizzamento logico di un processo è in generale diverso dai veri e propri indirizzi di memoria e quindi (generalmente a tempo di esecuzione) si rende necessario un meccanismo di traduzione {indirizzo logico}/{indirizzo fisico}, per poter accedere al dato necessario. Se ad esempio il calcolatore in questione implementa la sua memoria virtuale a pagine, un processo in esecuzione potrebbe riferire un dato contenuto nella pagina logica 3, posizione 100. Dal momento che tale pagina logica, sempreché sia presente in memoria, potrebbe essere locata nella pagina fisica 5, l'indirizzo effettivo per prelevare il dato sarà pagina 5 locazione 100.

Per attuare questa traduzione nel più breve tempo possibile (ogni accesso alla memoria deve essere tradotto) tale compito è interamente demandato ad una unità specializzata interposta tra processore e memoria denominata appunto MMU. Essa riceve l'indirizzo logico dal processore, esegue immediatamente la traduzione in indirizzo fisico, richiede il dato alla memoria e lo invia al processore che continua l'elaborazione normalmente, non essendosi accorto di nulla (per *lui* è stato un normale accesso in memoria).

Oltre a questo, una MMU che si rispetti si occupa anche di smistare indirizzamenti a periferiche I/O memory mapped, a segnalare eventuali fault di pagina o di segmento, a gestire le interruzioni (questo assieme al processore).

FF04	PCR D	D50B	VR
FF03	PCR C	D50A	P1 H
FF02	PCR B	D509	P1 L
FF01	PCR A	D508	PØ H
FF00	CR	D507	PØ L
		D506	RCR
		D505	MCR
		D504	PCR D
		D503	PCR C
		D502	PCR B
		D501	PCR A
		D500	CR

Figura 1 - I 12 registri della MMU.

L'MMU del 128

L'utilizzo di una MMU nel 128 non è certamente necessaria per i motivi sopra esposti. Trova la sua ragion d'essere dato che, tutti ormai lo sanno, il processore di questo è capace di indirizzare solo 64 k, mentre la memoria disponibile tra ram e rom è molta di più. Come nel caso dei *veri calcolatori*, avremo che un riferimento logico ad una cella di memoria è dato dalla coppia (banco, posizione) mentre l'indirizzamento fisico... beh, quello proprio non è identificabile dato che la memoria fisica del 128 è sparsa per tutta la macchina sottoforma di due banchi ram da 64 k l'uno, 16 k rom del sistema operativo, 32 k rom del Basic + monitor, generatore dei caratteri, memory mapped I/O ecc.

Purtroppo, a livello hardware, non è possibile che un programma locato in un banco possa fare riferimenti ad altri banchi, se non comandato alla MMU una commutazione di banco. Fortunatamente al livello di sistema operativo ciò non accade essendo disponibili delle apposite routine che permettono di accedere a qualunque locazione di qualsiasi banco semplicemente effettuando opportune chiamate (cfr. MC n. 57, 128 da zero).

12 registri

Per impartire ordini alla MMU, che come vedremo non si occupa solo dei banchi nudi e crudi, si utilizzano 12 re-

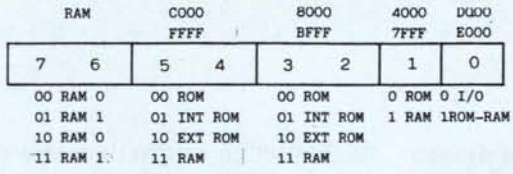

Figura 2 - Registro CR ($D500 - $FF00).

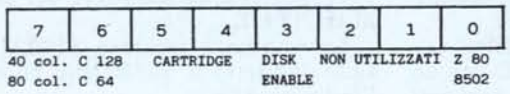

Figura 3 - Registro MCR ($D505).

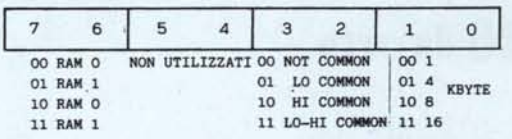

Figura 4 - Registro RCR ($D506).

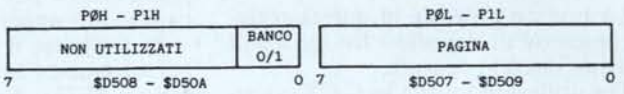

Figura 5 - Registri P0-P1.

gistri mappati a partire dall'indirizzo esadecimale $D500 del banco 15 (figura 1). In quella zona, come più volte ripetuto, è mappato l'I/O della macchina compreso quindi i registri per il suono, per i CIA, per il video ecc. I primi 5 registri della MMU sono inoltre mappati a partire dall'indirizzo esadecimale $FF00 di ogni banco: ciò per evitare che, una volta commutato un determinato banco di memoria, non sapremmo più come tornare indietro.

Del primo registro, $FF00 o $D500, ne abbiamo già parlato nel numero 57, ed è lì che vi rimandiamo per maggiori chiarimenti. Esso è la vera cloche di comando della memoria dato che settando o resettendo i suoi bit si può impostare qualsiasi configurazione, anche non prevista dal comando BANK del Basic. In figura 2 è mostrato tale registro e il significato dei suoi bit.

I registri 1-4, locati a $D501-$D504 del banco 14, e disponibili solo in lettura anche a $FF01..04, servono per impostare delle preconfigurazioni di memoria, quelle che più useremo, richiamabili semplicemente accedendo ai registri copia corrispondenti. Ovvero, una volta settate le nostre configurazioni preferite a partire da $D501, per effettuare una commutazione sarà sufficiente accedere in scrittura nel registro copia corrispondente ($FF01..04) ed essere così *catapultati* nella nuova configurazione di memoria.

Il primo dei registri non disponibili sottoforma di copia è registro Modo di Configurazione (MCR) ed è raffigurato in figura 3. In esso possiamo leggere alcune informazioni a dire il vero non troppo interessanti: ad esempio se all'accensione il tasto 40/80 colonne era premuto o meno. Il bit 0 sembra l'unico degno di nota dato che controlla quale processore è attualmente al lavoro (Z80 o 8502).

A partire dall'indirizzo $D506 le cose si fanno sempre più interessanti. Con questo primo registro (Registro Configurazione Ram, figura 4) è possibile configurare la memoria secondo *altri punti di vista.* Ad esempio possiamo cambiare la ram visibile dal Video Interface Chip (40 colonne) impostando il banco 1. È così possibile effettuare rapidi swap di schermo, sia in bassa che in alta risoluzione (oppure swap di sprite...) semplicemente allocando lo stesso spazio di memoria video sia nel banco 0 che nel banco 1. Per effettuare lo swap sarà sufficiente comunicare alla MMU quale banco deve essere visibile dal VIC e il gioco è fatto.

Sempre nel registro RCR troviamo la possibilità di definire aree comuni ai due banchi in testa o in coda, di dimensioni pari a 1,4,8 o 16 k byte. Per default, come detto sempre alcuni numeri fa, l'area di memoria comune assomma a 1 k, allocato a inizio memoria. Grazie a questo artificio, un programma *giacente* in una zona di memoria comune può ordinare commutazioni di configurazione alla MMU senza perdere il controllo del flusso.

Puntatori di pagina

Tramite la MMU del 128 è possibile definire pagine (non banchi, attenzione) 0 e 1 in qualsiasi punto della memoria del 128. Come si sa, il processore 8502 permette alcuni modi di indirizzamento solo in pagina 0 mentre lo stack di sistema è sempre allocato in pagina 1. Ad esempio, per spostare grosse aree di memoria, chiunque abbia usato solo un po' il linguaggio macchina, conoscerà il modo di indirizzamento:

```
LDA ($PP), Y
```

dove $PP è un indirizzo in pagina 0. Chi invece il linguaggio macchina lo usa spesso e volentieri, avrà notato come le locazioni libere in pagina 0 sono sempre poche (sono quasi tutte adoperate dal sistema) e occorre ricorrere a vari artifizi per rubarne qualcuna in più. Utilizzando opportunamente la MMU possiamo tagliare la testa al toro definendo una nuova pagina 0, ad esempio a partire dall'indirizzo $1000 e disporre così di 256 locazioni di tale tipo, tutte libere per noi. Ovvero, dopo aver impostato opportunamente la MMU, scrivendo:

```
STA $03
```

immetteremo il contenuto dell'accumulatore nella locazione $1003 e un accesso del tipo:

```
LDA ($03), Y
```

equivale a un LDA ($1003), Y addirittura non disponibile normalmente.

Come detto prima, è possibile fare lo stesso giochetto anche per lo stack (pagina 1) nel qual caso potremmo implementarne uno nuovo in qualsiasi punto della memoria. Ciò può essere utile non tanto come nuovo stack, ma come indirizzamento rapido di una qualsiasi area di memoria. Ad esempio per azzerare 256 byte a partire da $1000 allochiamo lì in nostro nuovo stack e, caricato nell'accumulatore il valore 0, non ci resta che dare 256 PHA per essere accontentati. Si noti che un «PHA» è ben più rapido di un normale «STA 1000, X» dato che il primo richiede 3 cicli di clock il secondo 5. In figura 5 sono mostrati i registri interessati, per la pagina 0 e 1. Le rispettive parti basse indicano la pagina riferita come pagina 0 o 1, mentre delle parti alte interessa solo il bit meno significativo nel quale indicheremo se ci riferiamo al banco 0 o 1 della ram.

Attenzione a rimettere a posto stack, stack pointer e pagina 0 dopo l'uso: avremmo sicuramente effetti catastrofici dimenticandocene.

Per finire, in figura 6, è mostrato il registro Versione, nel quale possiamo leggere (!) quanti banchi ram possiede il nostro 128 e, addirittura!, la versione della nostra MMU. Il massimo.

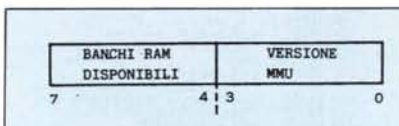

Figura 6- Registro VR ($D50B).