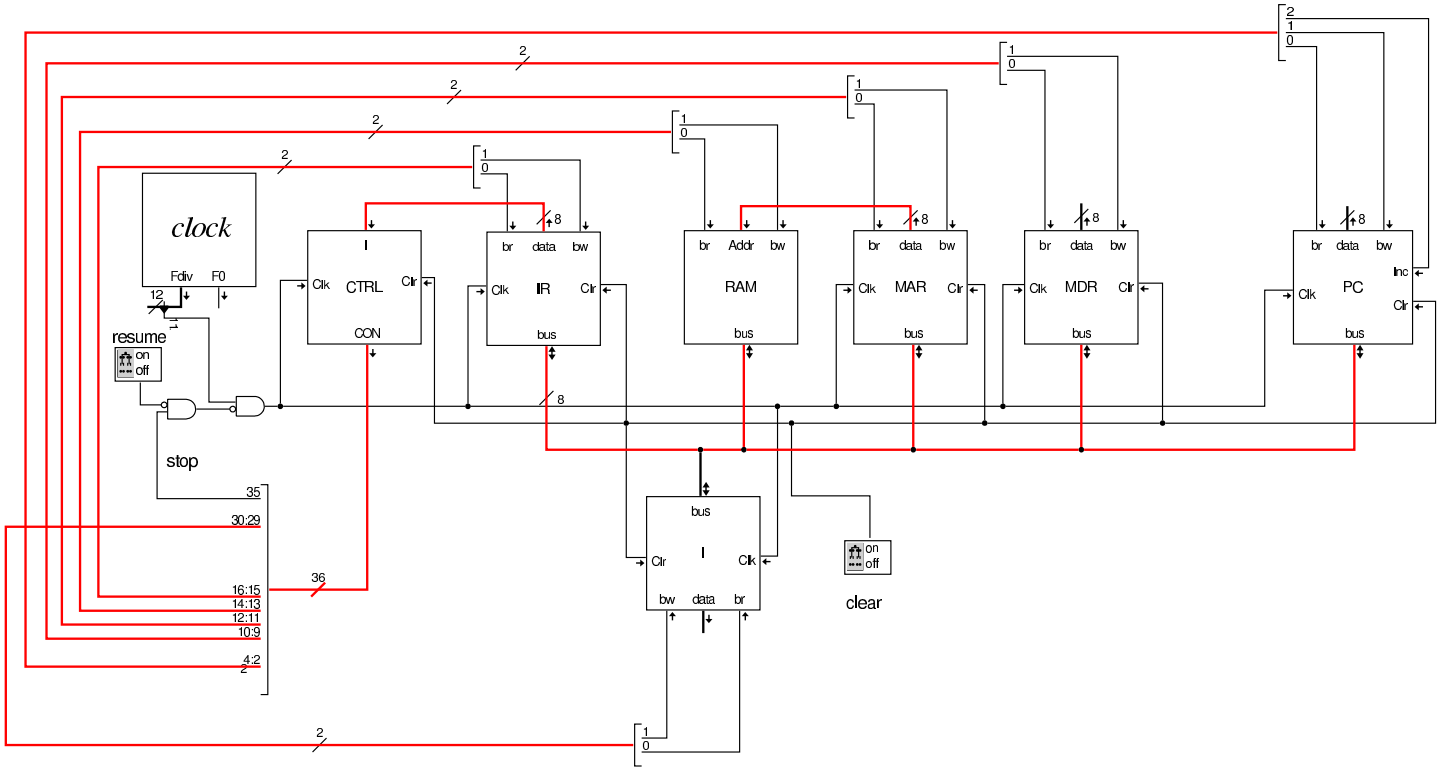
Versione B: indice della memoria

Nella seconda versione della CPU dimostrativa, viene aggiunto soltanto un registro speciale, denominato ***I***, il cui scopo è quello di contenere un indice della memoria. Nello specifico, serve a poter leggere o scrivere nella memoria RAM, attraverso un indice che possa essere gestito. Il registro ***I*** è realizzato nello stesso modo di ***MDR***, ***MAR*** e ***IR***.

Figura u107.1. Il bus della CPU nella sua seconda fase realizzativa.



Nel codice che descrive i campi del bus di controllo, si aggiungono quelli seguenti, i quali servono specificatamente a gestire il registro

I:

```
field i_br[29];          // I <-- bus
field i_bw[30];          // I --> bus
```

Nell'elenco dei codici operativi si aggiungono istruzioni nuove e lo stesso poi nella descrizione del microcodice:

```
op load_imm
{
  map load_imm : 1;                  // load from address #nn
  +0[7:0]=1;
  operands op_1;
};
op load_reg
{
  map load_reg : 2;                  // load from address %I
  +0[7:0]=2;
  operands op_0;
};
op store_imm {
  map store_imm : 3;                 // store to address #nn
  +0[7:0]=3;
  operands op_1;
};
op store_reg {
  map store_reg : 4;                 // store to address I
  +0[7:0]=4;
  operands op_0;
};
op move_mdr_i {
  map move_mdr_i : 11;               // move MDR to I
  +0[7:0]=11;
  operands op_0;
};
op move_i_mdr {
```

```
  map move_i_mdr : 12;                    // move I to MDR
  +0[7:0]=12;
  operands op_0;
};
```

```
begin microcode @ 0
…
load_imm:
    mar_br pc_bw;                         // MAR <-- PC
    pc_Inc;                               // PC++
    // La memoria non ha un clock,
    // quindi, non si può passare
    // direttamente a MAR.
    i_br ram_bw;                          // I <-- RAM[MAR]
    mar_br i_bw;                          // MAR <-- I
    mdr_br ram_bw;                        // MDR <-- RAM[MAR]
    ctrl_start ctrl_load;                 // CNT <-- 0
//
load_reg:
    mar_br i_bw;                          // MAR <-- I
    mdr_br ram_bw;                        // MDR <-- RAM[MAR]
    ctrl_start ctrl_load;                 // CNT <-- 0
//
store_imm:
    mar_br pc_bw;                         // MAR <-- PC
    pc_Inc;                               // PC++
    i_br ram_bw;                          // I <-- RAM[MAR]
    mar_br i_bw;                          // MAR <-- I
    ram_br mdr_bw;                        // RAM[MAR] <-- MDR
    ctrl_start ctrl_load;                 // CNT <-- 0
//
store_reg:
    mar_br i_bw;                          // MAR <-- I
    ram_br mdr_bw;                        // RAM[MAR] <-- MDR
```

```
    ctrl_start ctrl_load;        // CNT <-- 0
//
move_mdr_i:
    i_br mdr_bw;                 // I <-- MDR
    ctrl_start ctrl_load;        // CNT <-- 0
//
move_i_mdr:
    mdr_br i_bw;                 // MDR <-- I
    ctrl_start ctrl_load;        // CNT <-- 0
…
end
```

Figura u107.5. Corrispondenza con il contenuto della memoria che rappresenta il microcodice (la coppia ***m1*** e ***m2*** dell'unità di controllo).

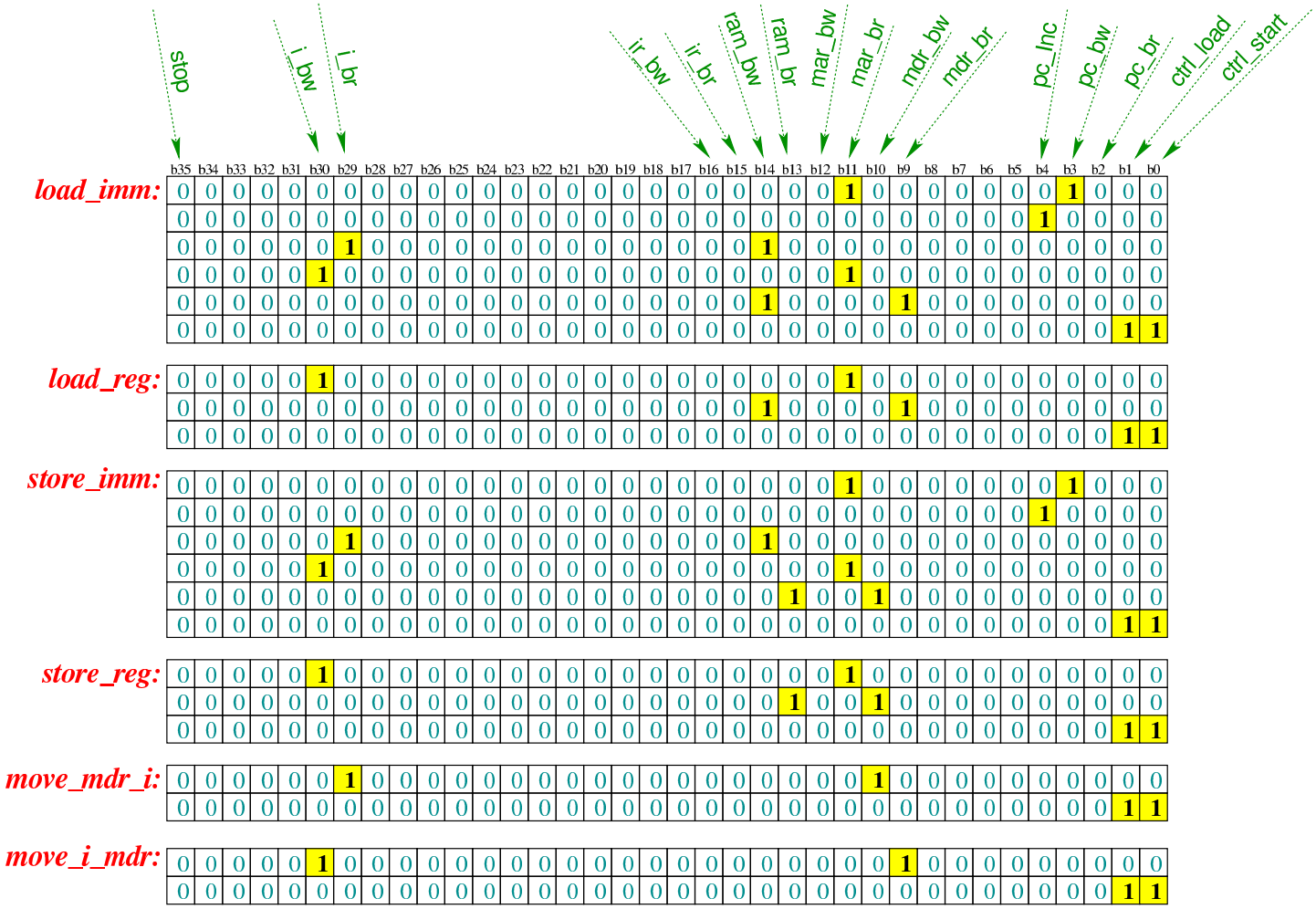


Tabella u107.6. Elenco delle macroistruzioni aggiunte in questa versione della CPU dimostrativa.

Sintassi	Descrizione
`load_imm` ***indirizzo***	*load immediate*: carica nel registro ***MDR*** il contenuto della cella di memoria che corrisponde all'indirizzo indicato dall'argomento. Contestualmente, il registro ***I*** viene modificato e alla fine contiene l'indirizzo di memoria in questione.

Sintassi	Descrizione
`load_reg`	*load register*: carica nel registro ***MDR*** il contenuto della cella di memoria che corrisponde all'indirizzo indicato dal valore contenuto nel registro ***I***.
`store_imm` ***indirizzo***	*store immediate*: salva in memoria, all'indirizzo specificato come argomento, il valore contenuto nel registro ***MDR***. Contestualmente, il registro ***I*** viene modificato e alla fine contiene l'indirizzo di memoria in questione.
`store_reg`	*store register*: salva in memoria, all'indirizzo specificato dal registro ***I***, il valore contenuto nel registro ***MDR***.
`move_mdr_i`	Copia il contenuto del registro ***MDR*** nel registro ***I***.
`move_i_mdr`	Copia il contenuto del registro ***I*** nel registro ***MDR***.

Istruzioni «load»

«

Come primo esempio viene proposto il macrocodice seguente:

```
begin macrocode @ 0
start:
        load_imm #data_1
        move_mdr_i
        load_reg
stop:
        stop
data_1:
       .byte 3
end
```

In pratica, viene caricato nel registro ***MDR*** il valore corrispondente all'indirizzo in cui si trova l'etichetta '**`data_1:`**' (facendo i conti si tratta dell'indirizzo 5); successivamente, il valore di ***MDR*** viene copiato nel registro ***I*** e quindi viene caricato nel registro ***MDR*** quanto contenuto nell'indirizzo di memoria corrispondente al valore di ***I***: dal momento che a quel indirizzo si trova il valore 2, corrispondente al codice operativo dell'istruzione **`load_reg`**, al termine, il registro ***MDR*** contiene tale valore. Il file completo che descrive le memorie per Tkgate dovrebbe essere disponibile all'indirizzo *allegati/circuiti-logici/scpu-sub-b.gm*

Figura u107.8. Contenuto della memoria RAM. Le celle indicate con «xx» hanno un valore indifferente.

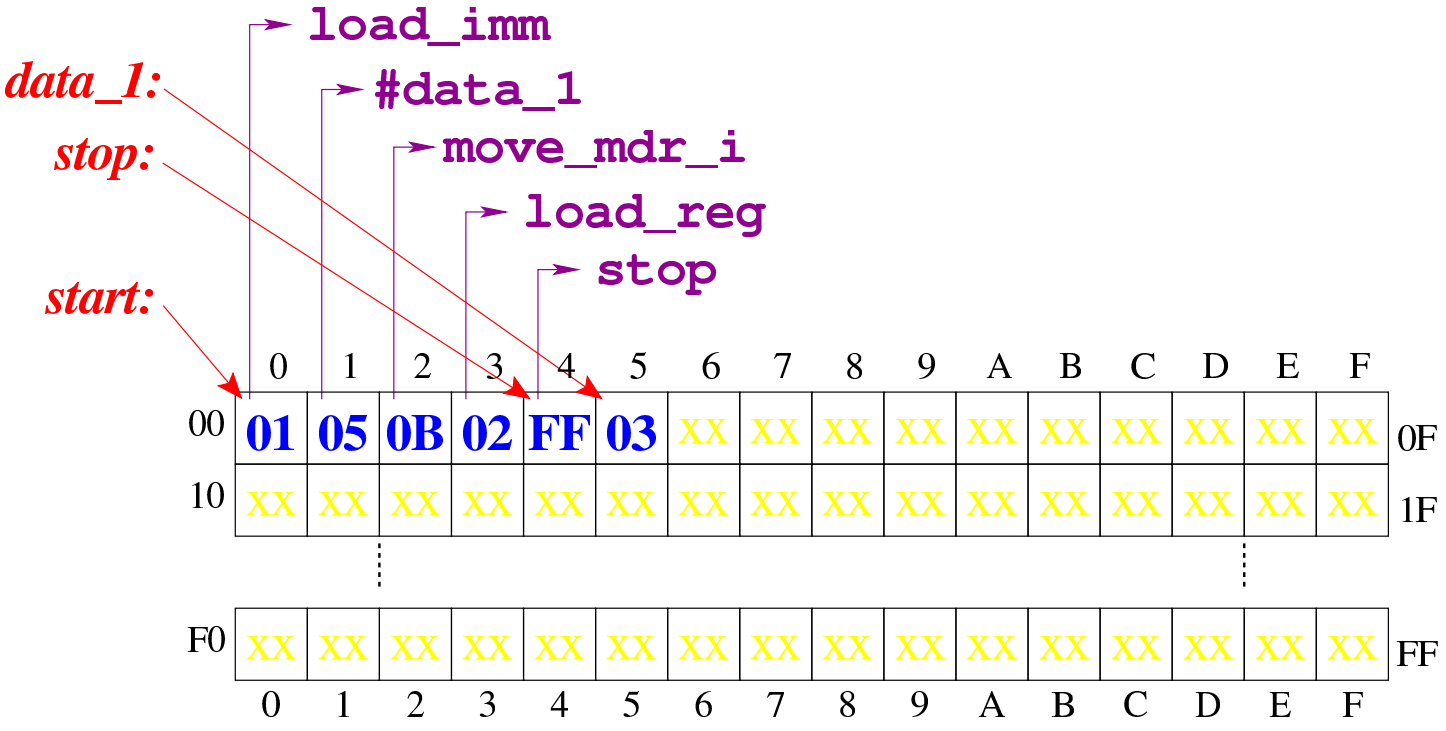
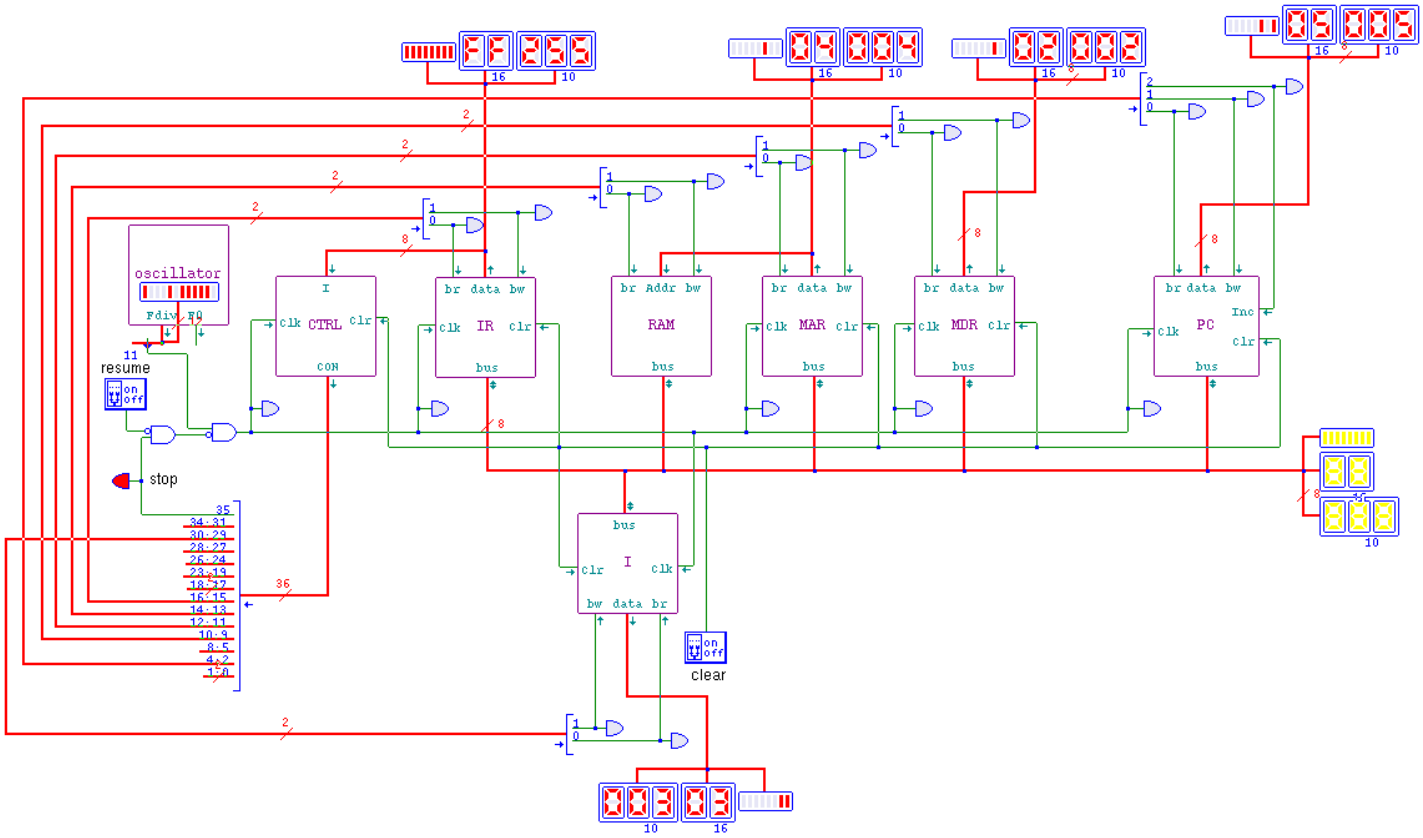


Figura u107.9. Situazione conclusiva del bus dati, dopo l'esecuzione delle istruzioni di caricamento. Video: *http://www.youtube.com/watch?v=AXUSrH49cF49w*



Istruzioni «store»

«

Viene proposto un altro esempio di macrocodice, nel quale si sperimentano le istruzioni **`store_imm`** e **`store_reg`**:

```
begin macrocode @ 0
start:
        load_imm #data_1
        store_imm #data_2
        move_mdr_i
        store_reg
stop:
        stop
data_1:
```

```
        .byte 15
data_2:
        .byte 0
end
```

In questo caso, si carica nel registro ***MDR*** il valore contenuto in memoria in corrispondenza dell'etichetta '**`data_1:`**'; quindi si memorizza, in corrispondenza della posizione di memoria corrispondente all'etichetta '**`data_2:`**', il valore contenuto in ***MDR*** (in pratica, in quella destinazione che prima conteneva il valore zero, viene copiato il valore 15, ovvero $0F_{16}$); quindi il contenuto del registro ***MDR*** viene copiato nel registro ***I*** e poi viene memorizzato il contenuto di ***MDR*** (che è rimasto sempre 15) nella posizione di memoria corrispondente al valore del registro ***I***. In pratica, alla fine si va a scrivere anche nella posizione 15 ($0F_{16}$) della memoria, e ci si mette il valore 15.

Figura u107.11. Contenuto della memoria RAM all'inizio dell'esecuzione.

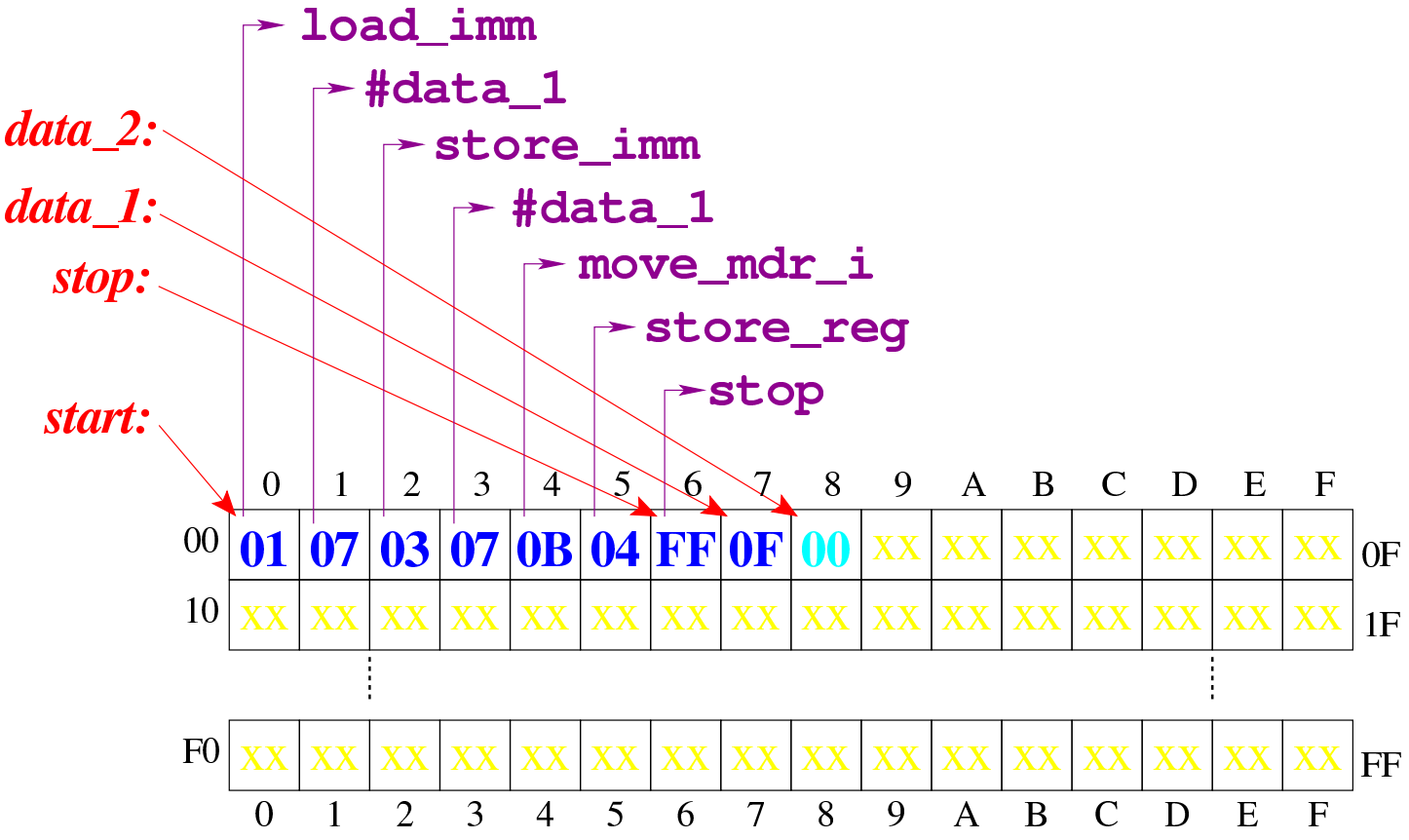


Figura u107.12. Contenuto della memoria RAM dopo l'esecuzione dell'istruzione **`store_imm`**.

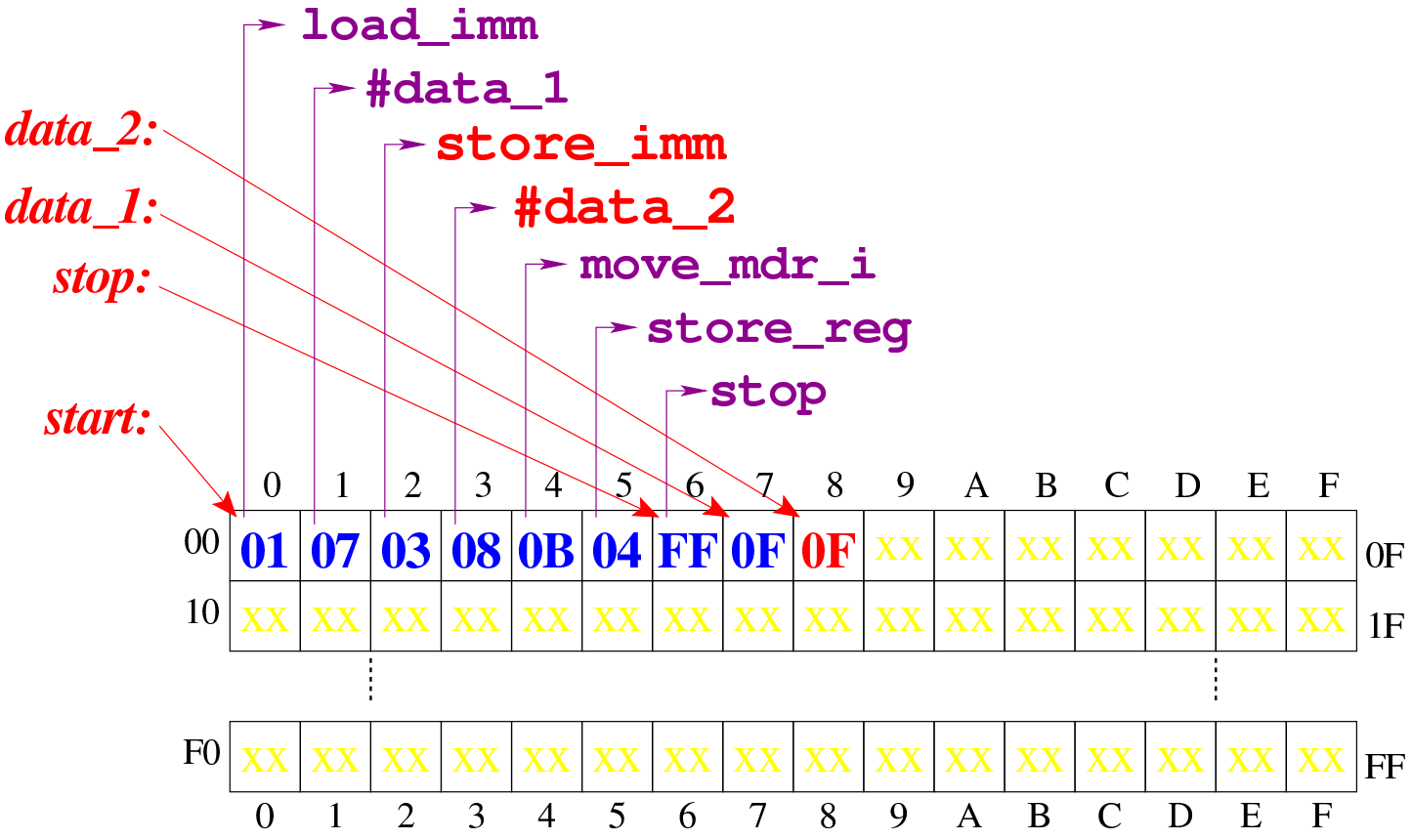


Figura u107.13. Contenuto della memoria RAM al termine dell'esecuzione.

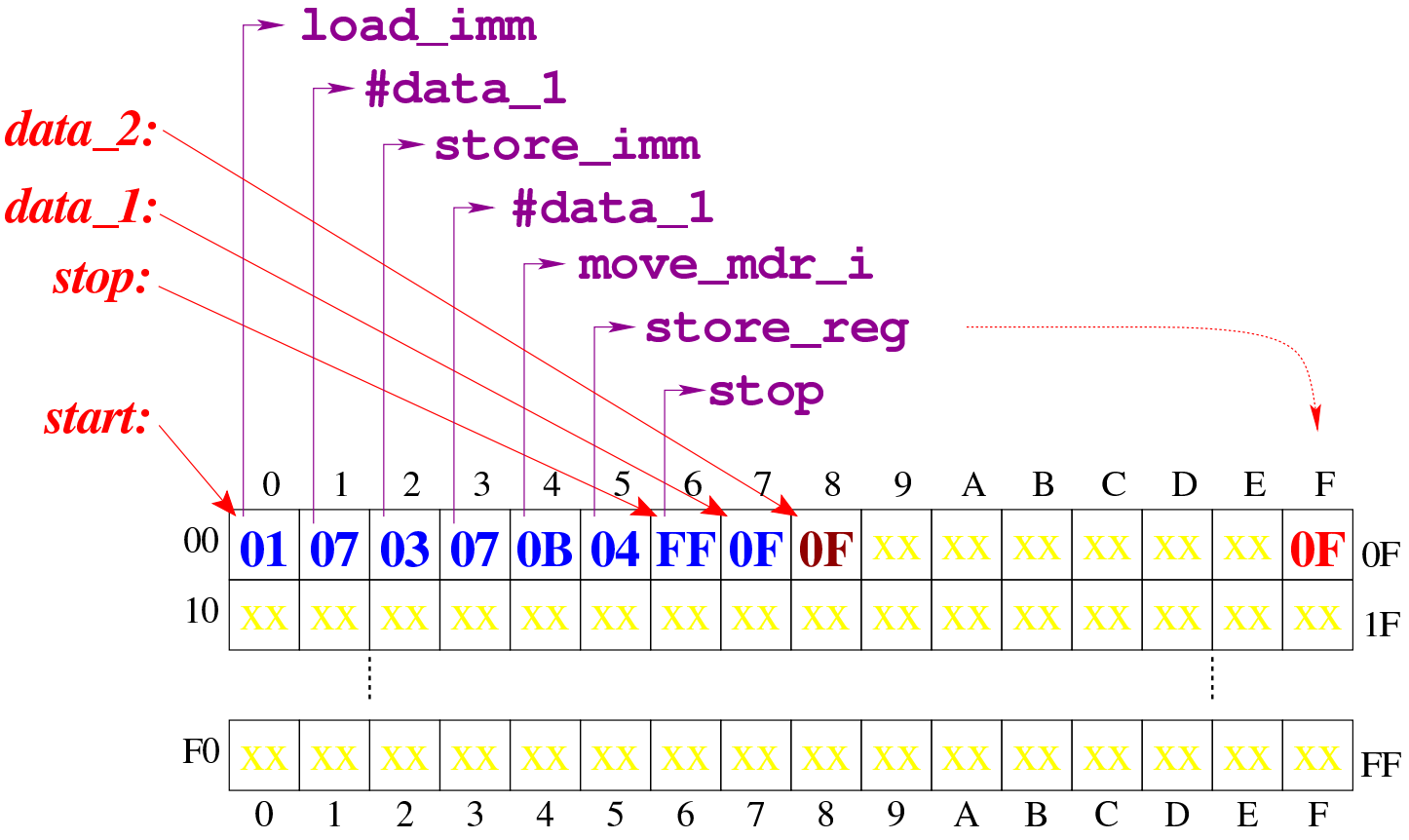
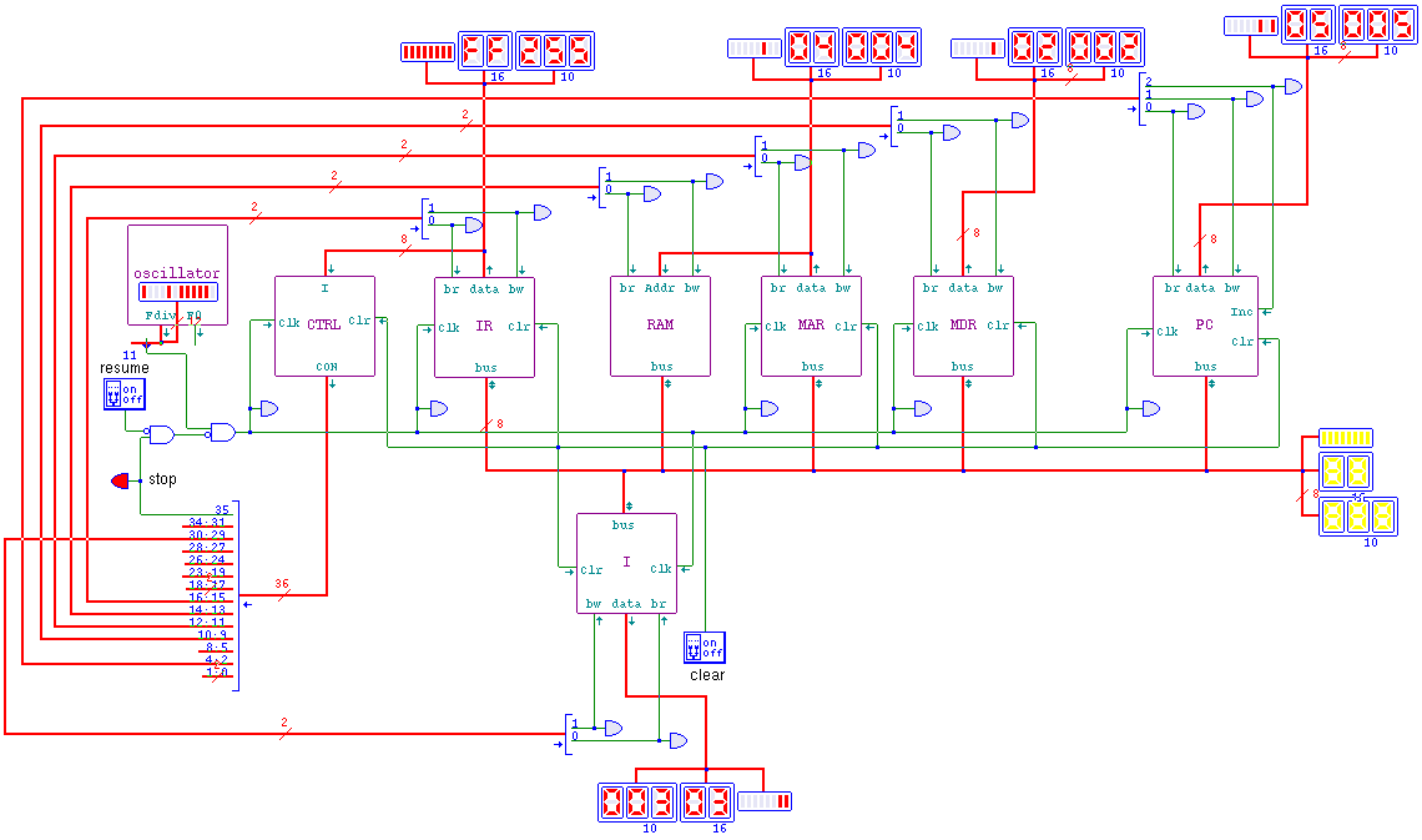


Figura u107.14. Situazione conclusiva del bus dati, dopo l'esecuzione delle istruzioni di memorizzazione. Video: *http://www.youtube.com/watch?v=lHxx3SR56hE56*



1916