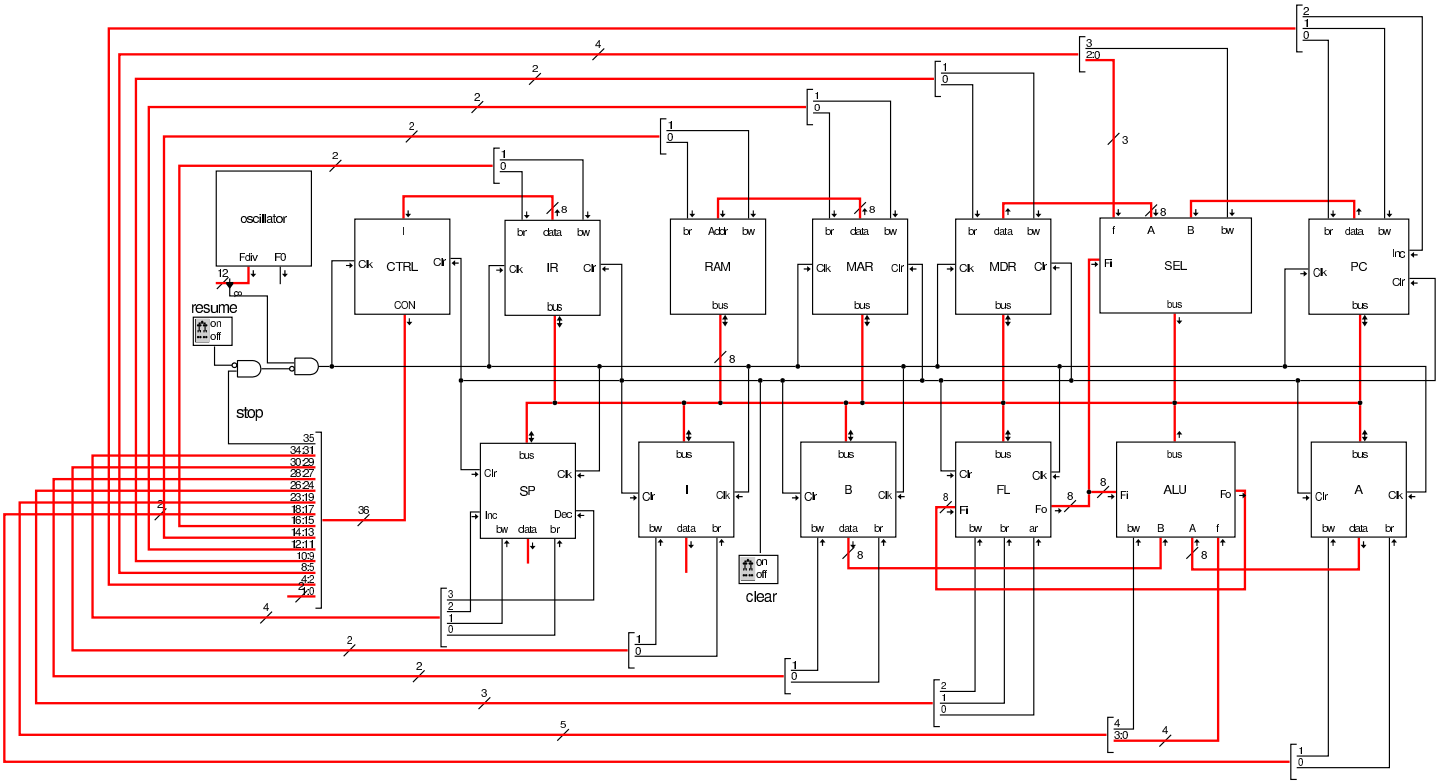
Versione G: pila

Nella settima versione della CPU dimostrativa, viene aggiunto il registro ***SP*** (*stack pointer*), utilizzato come indice per la pila dei dati. La pila serve principalmente a consentire le chiamate di procedure, tramite istruzioni **`call`** e **`return`**, oltre che a poter salvare e recuperare lo stato dei altri registri.

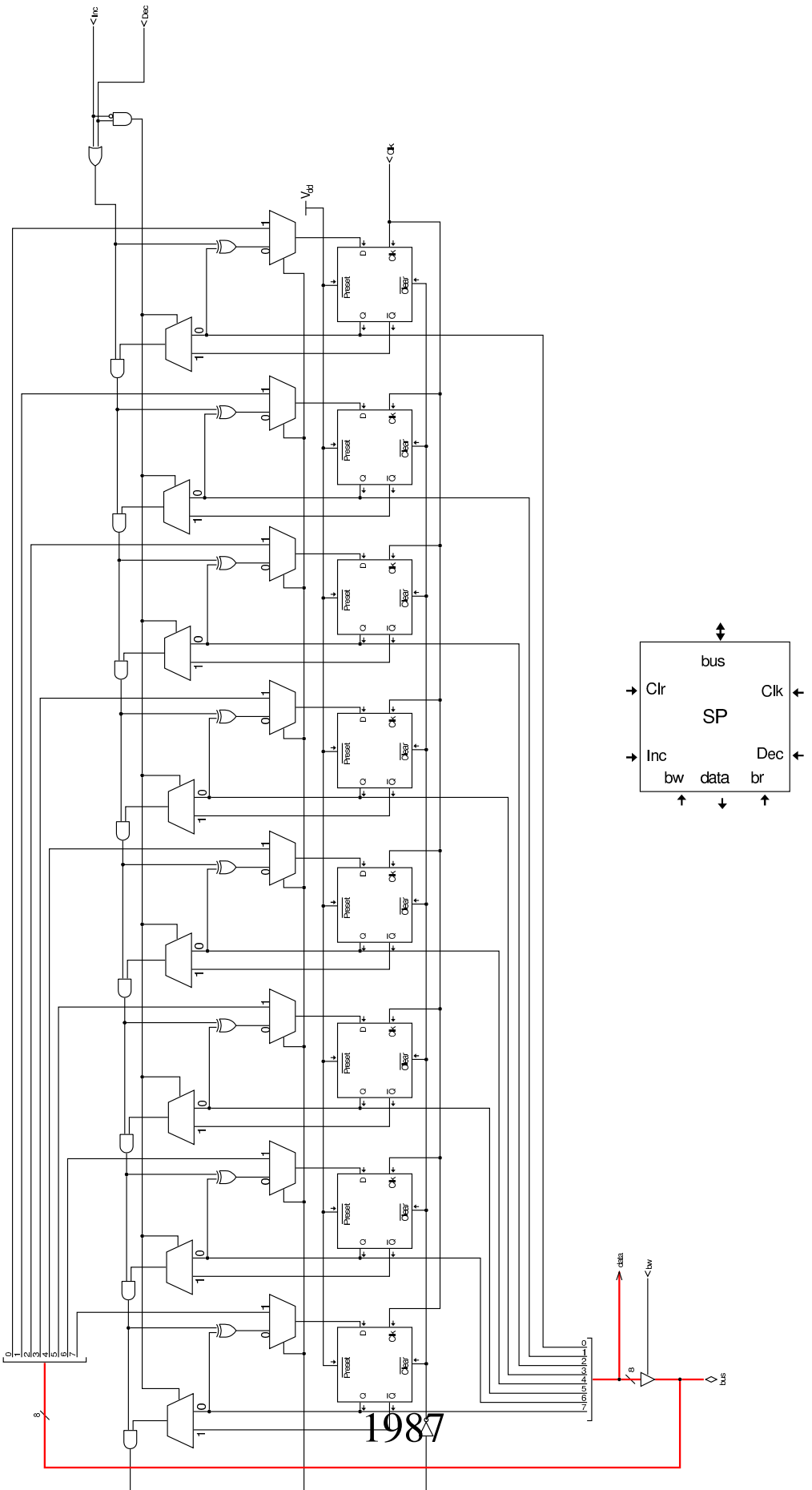
Figura u112.1. Il bus della CPU con l'aggiunta del registro ***SP*** per la gestione della pila.



Il registro ***SP*** ha due ingressi supplementari, ***Inc*** e ***Dec***, con lo scopo, rispettivamente, di incrementare o diminuire il valore

memorizzato nel registro stesso, di una unità.

Figura u112.2. La struttura interna del registro **SP**.



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

```
field sp_br[31];          // SP <-- bus
field sp_bw[32];          // SP --> bus
field sp_Inc[33];         // SP++
field sp_Dec[34];         // SP--
```

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

```
op call_imm {
  map call_imm : 24;                    // call #nn
  +0[7:0]=24;
  operands op_1;
};
op call_reg {
  map call_reg : 25;                    // call I
  +0[7:0]=25;
  operands op_0;
};
op return {
  map return : 26;                      // return
  +0[7:0]=26;
  operands op_0;
};
op push_mdr {
  map push_mdr : 27;                    // push MDR
  +0[7:0]=27;
  operands op_0;
};
op push_a {
  map push_a : 28;                      // push A
  +0[7:0]=28;
  operands op_0;
```

```
};
op push_b {
  map push_b : 29;                          // push B
  +0[7:0]=29;
  operands op_0;
};
op push_fl {
  map push_fl : 30;                         // push FL
  +0[7:0]=30;
  operands op_0;
};
op push_i {
  map push_i : 31;                          // push I
  +0[7:0]=31;
  operands op_0;
};
op pop_mdr {
  map pop_mdr : 48;                         // pop MDR
  +0[7:0]=48;
  operands op_0;
};
op pop_a {
  map pop_a : 49;                           // pop A
  +0[7:0]=49;
  operands op_0;
};
op pop_b {
  map pop_b : 50;                           // pop B
  +0[7:0]=50;
  operands op_0;
};
op pop_fl {
  map pop_fl : 51;                          // pop FL
  +0[7:0]=51;
  operands op_0;
```

```
};
op pop_i {
  map pop_i : 52;                          // pop I
  +0[7:0]=52;
  operands op_0;
};
```

```
begin microcode @ 0
…
call_imm:
    mar_br pc_bw;                          // MAR <-- PC
    pc_Inc;                                // PC++
    mdr_br ram_bw;                         // MDR <-- RAM[mar]
    sp_Dec;                                // SC--
    mar_br sp_bw;                          // MAR <-- SP
    ram_br pc_bw;                          // RAM[mar] <-- PC
    pc_br mdr_bw;                          // PC <-- MDR
    ctrl_start ctrl_load;                  // CNT <-- 0
//
call_reg:
    sp_Dec;                                // SP--
    mar_br sp_bw;                          // MAR <-- SP
    ram_br pc_bw;                          // RAM[mar] <-- PC
    pc_br i_bw;                            // PC <-- I
    ctrl_start ctrl_load;                  // CNT <-- 0
//
return:
    mar_br sp_bw;                          // MAR <-- SP
    sp_Inc;                                // SP++;
    pc_br ram_bw;                          // PC <-- RAM[mar]
    ctrl_start ctrl_load;                  // CNT <-- 0
//
push_mdr:
    sp_Dec;                                // SP--
    mar_br sp_bw;                          // MAR <-- SP
    ram_br mdr_bw;                         // RAM[mar] <-- MDR
    ctrl_start ctrl_load;                  // CNT <-- 0
//
push_a:
    sp_Dec;                                // SP--
```

```
    mar_br sp_bw;                   // MAR <-- SP
    ram_br a_bw;                    // RAM[mar] <-- A
    ctrl_start ctrl_load;           // CNT <-- 0
//
push_b:
    sp_Dec;                         // SP--
    mar_br sp_bw;                   // MAR <-- SP
    ram_br b_bw;                    // RAM[mar] <-- B
    ctrl_start ctrl_load;           // CNT <-- 0
//
push_fl:
    sp_Dec;                         // SP--
    mar_br sp_bw;                   // MAR <-- SP
    ram_br fl_bw;                   // RAM[mar] <-- FL
    ctrl_start ctrl_load;           // CNT <-- 0
//
push_i:
    sp_Dec;                         // SP--
    mar_br sp_bw;                   // MAR <-- SP
    ram_br i_bw;                    // RAM[mar] <-- I
    ctrl_start ctrl_load;           // CNT <-- 0
//
pop_mdr:
    mar_br sp_bw;                   // MAR <-- SP
    sp_Inc;                         // SP++
    mdr_br ram_bw;                  // MDR <-- RAM[mar]
    ctrl_start ctrl_load;           // CNT <-- 0
//
pop_a:
    mar_br sp_bw;                   // MAR <-- SP
    sp_Inc;                         // SP++
    a_br ram_bw;                    // A <-- RAM[mar]
    ctrl_start ctrl_load;           // CNT <-- 0
//
pop_b:
    mar_br sp_bw;                   // MAR <-- SP
    sp_Inc;                         // SP++
    b_br ram_bw;                    // B <-- RAM[mar]
    ctrl_start ctrl_load;           // CNT <-- 0
//
pop_fl:
```

```
    mar_br sp_bw;                    // MAR <-- SP
    sp_Inc;                          // SP++
    fl_br ram_bw;                    // FL <-- RAM[mar]
    ctrl_start ctrl_load;            // CNT <-- 0
//
pop_i:
    mar_br sp_bw;                    // MAR <-- SP
    sp_Inc;                          // SP++
    i_br ram_bw;                     // I <-- RAM[mar]
    ctrl_start ctrl_load;            // CNT <-- 0
…
end
```

Figura u112.6. Corrispondenza con il contenuto della memoria che rappresenta il microcodice (la coppia ***m1*** e ***m2*** dell'unità di controllo).

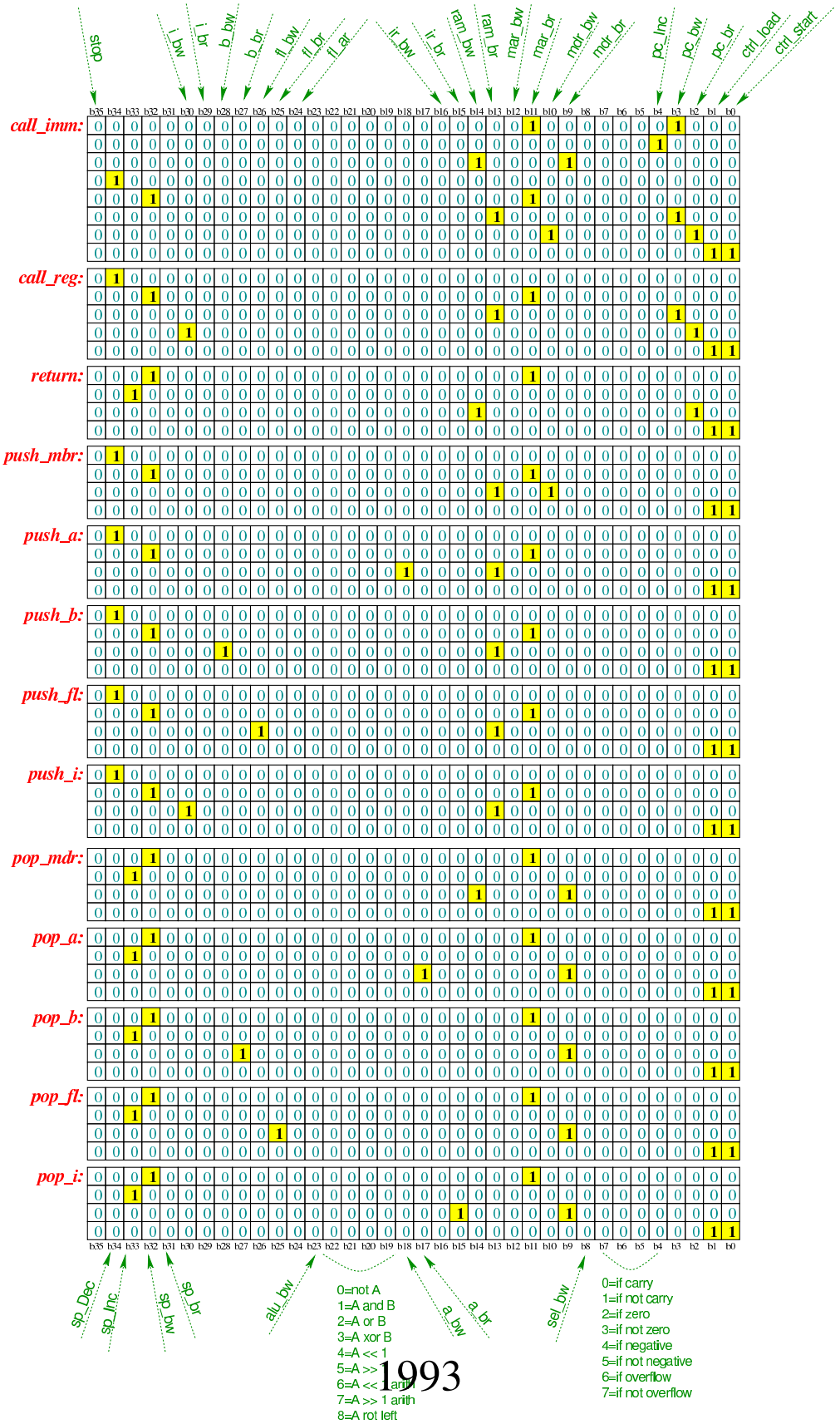


Tabella u112.7. Elenco delle macroistruzioni aggiunte in questa versione della CPU dimostrativa.

Sintassi	Descrizione
`call_imm` ***indirizzo***	Decrementa il registro ***SP*** e annota nella posizione di memoria a cui questo punta l'indirizzo dell'istruzione successiva, quindi aggiorna il registro ***PC*** con l'indirizzo indicato come argomento. In pratica esegue la chiamata di una procedura, annotando nella pila dei dati l'indirizzo di ritorno.
`call_reg`	Decrementa il registro ***SP*** e annota nella posizione di memoria a cui questo punta l'indirizzo dell'istruzione successiva, quindi aggiorna il registro ***PC*** con l'indirizzo contenuto nel registro ***I***. In pratica esegue la chiamata di una procedura, annotando nella pila dei dati l'indirizzo di ritorno.
`return`	Incrementa il registro ***SP*** e preleva l'indirizzo annotato in memoria, corrispondente all'indice della pila, quindi aggiorna il registro ***PC*** con tale valore. In pratica, consente di concludere una chiamata precedente.
`push_mdr`	Inserisce nella pila il valore del registro ***MDR***.
`push_a`	Inserisce nella pila il valore del registro ***A***.
`push_b`	Inserisce nella pila il valore del registro ***B***.
`push_fl`	Inserisce nella pila il valore del registro ***FL***.

Sintassi	Descrizione
`push_i`	Inserisce nella pila il valore del registro ***I***.
`pop_mdr`	Recupera dalla pila il valore del registro ***MDR***.
`pop_a`	Recupera dalla pila il valore del registro ***A***.
`pop_b`	Recupera dalla pila il valore del registro ***B***.
`pop_fl`	Recupera dalla pila il valore del registro ***FL***.
`pop_i`	Recupera dalla pila il valore del registro ***I***.

Istruzioni «push» e «pop»

«

Listato u112.8. Macrocodice per sperimentare l'uso delle istruzioni di inserimento ed estrazione di valori dalla pila dei dati: in questo caso, viene inserito nella pila il valore contenuto nel registro ***A*** e poi ripescato, ma nel registro ***B***. Il file completo che descrive le memorie per Tkgate dovrebbe essere disponibile presso *allegati/circuiti-logici/scpu-sub-g-push-pop.gm* .

```
begin macrocode @ 0
start:
        load_imm #sp_bottom
        move_mdr_sp
        load_imm #data_0
        move_mdr_a
        push_a
        pop_b

stop:
```

```
        stop
sp_bottom:
        .byte 0x10
data_0:
        .byte 0xCC
end
```

Figura u112.9. Contenuto della memoria RAM: la cella nella posizione $0F_{16}$ viene scritta dall'istruzione **`push_a`**. Le celle indicate con «xx» hanno un valore indifferente.

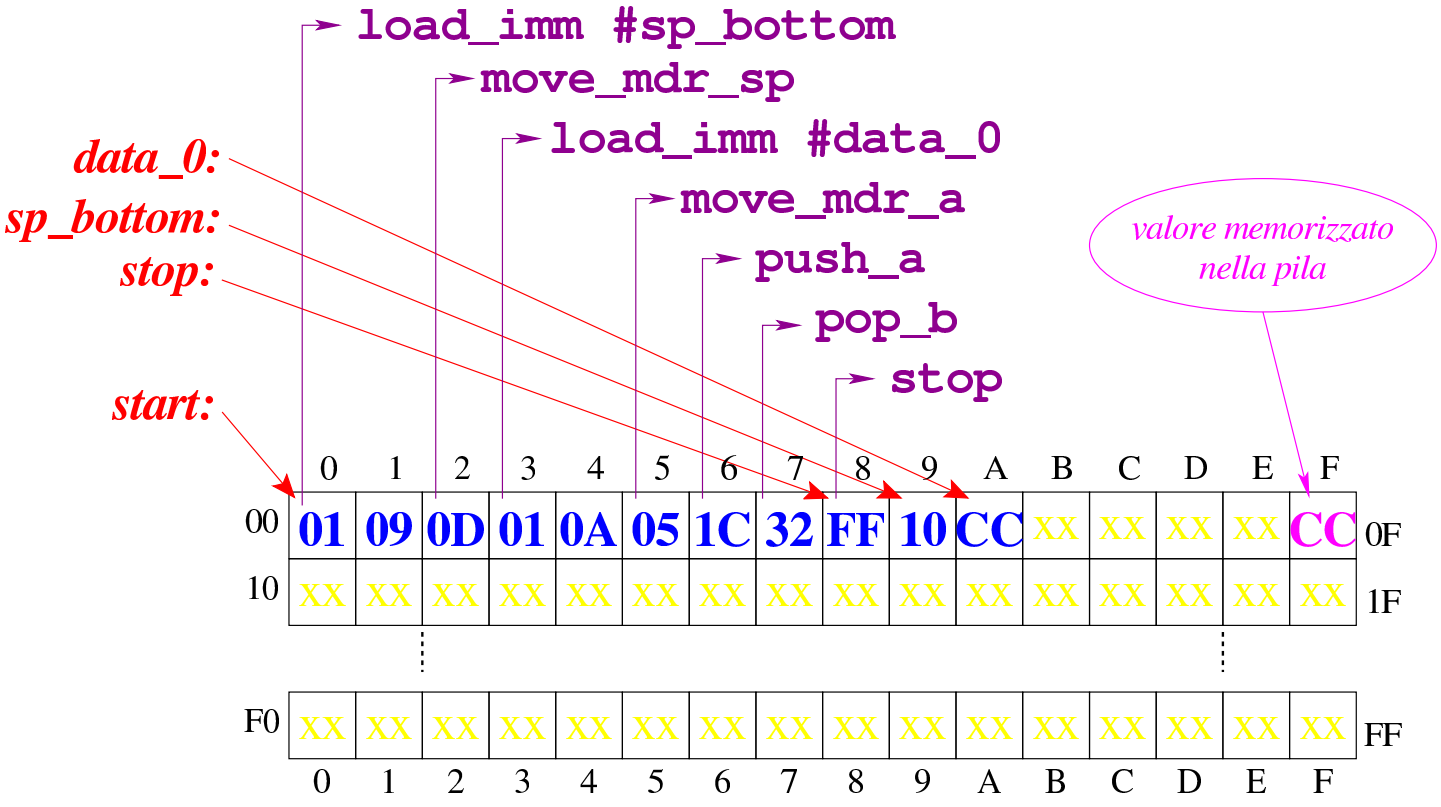
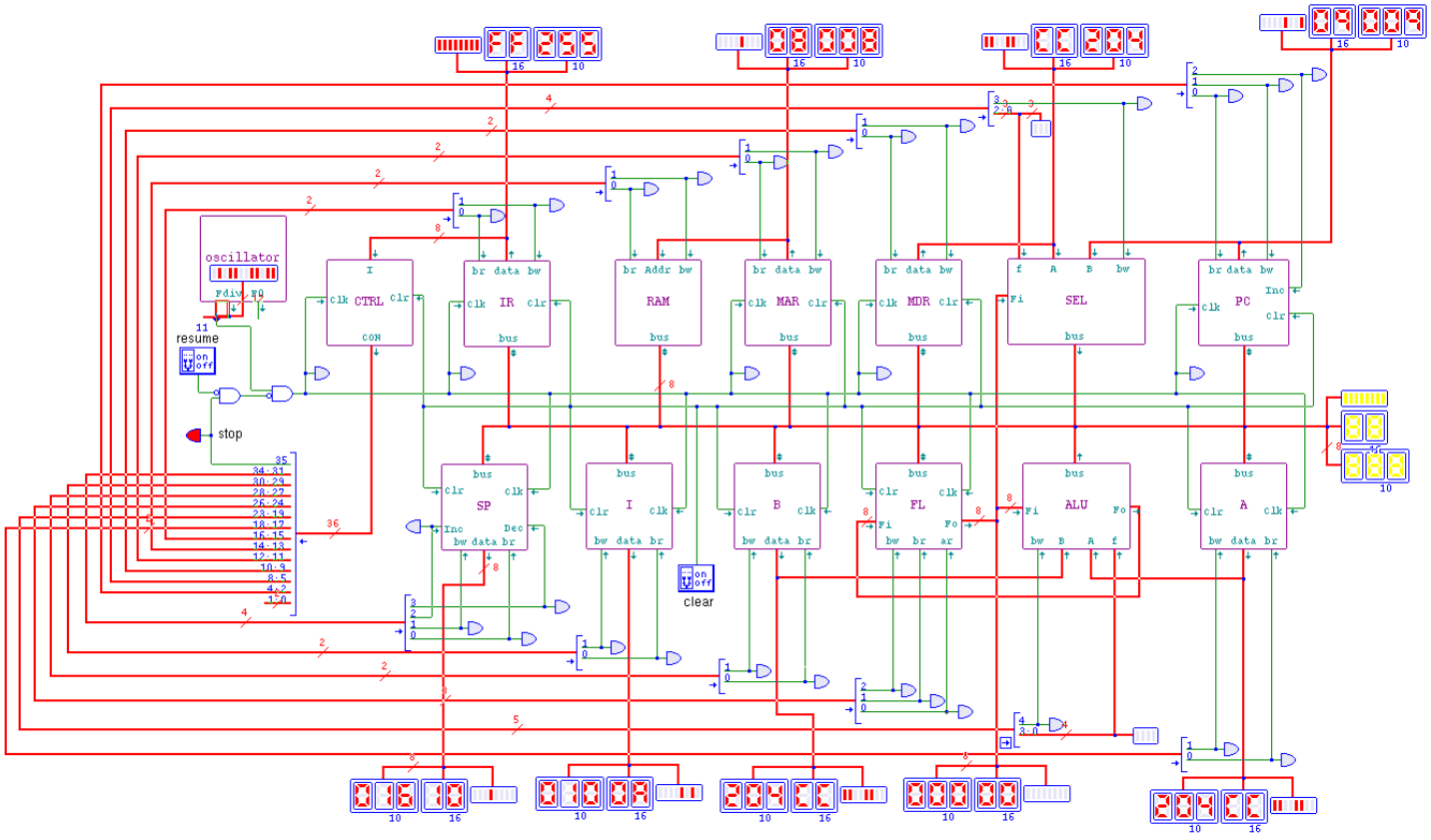


Figura u112.10. Situazione conclusiva del bus dati. Video: *http://www.youtube.com/watch?v=vAdVww3lD7I*



Istruzioni «call» e «return»

Listato u112.11. Macrocodice per sperimentare l'uso delle istruzioni di chiamata e ritorno dalle procedure. Il file completo che descrive le memorie per Tkgate dovrebbe essere disponibile presso *allegati/circuiti-logici/scpu-sub-g-call-return.gm* .

```
begin macrocode @ 0
start:
        load_imm #sp_bottom
        move_mdr_sp
        call_imm #elabora
        move_a_mdr
        move_mdr_b
        jump #stop
```

```
elabora:
        load_imm #data_0
        move_mdr_a
        load_imm #data_1
        move_mdr_b
        add
        return
stop:
        stop
sp_bottom:
        .byte 0x20
data_0:
        .byte 0x0A
data_1:
        .byte 0x0B
end
```

Figura u112.12. Contenuto della memoria RAM: la cella nella posizione $1F_{16}$ viene scritta dall'istruzione **`call_imm`**. Le celle indicate con «xx» hanno un valore indifferente.

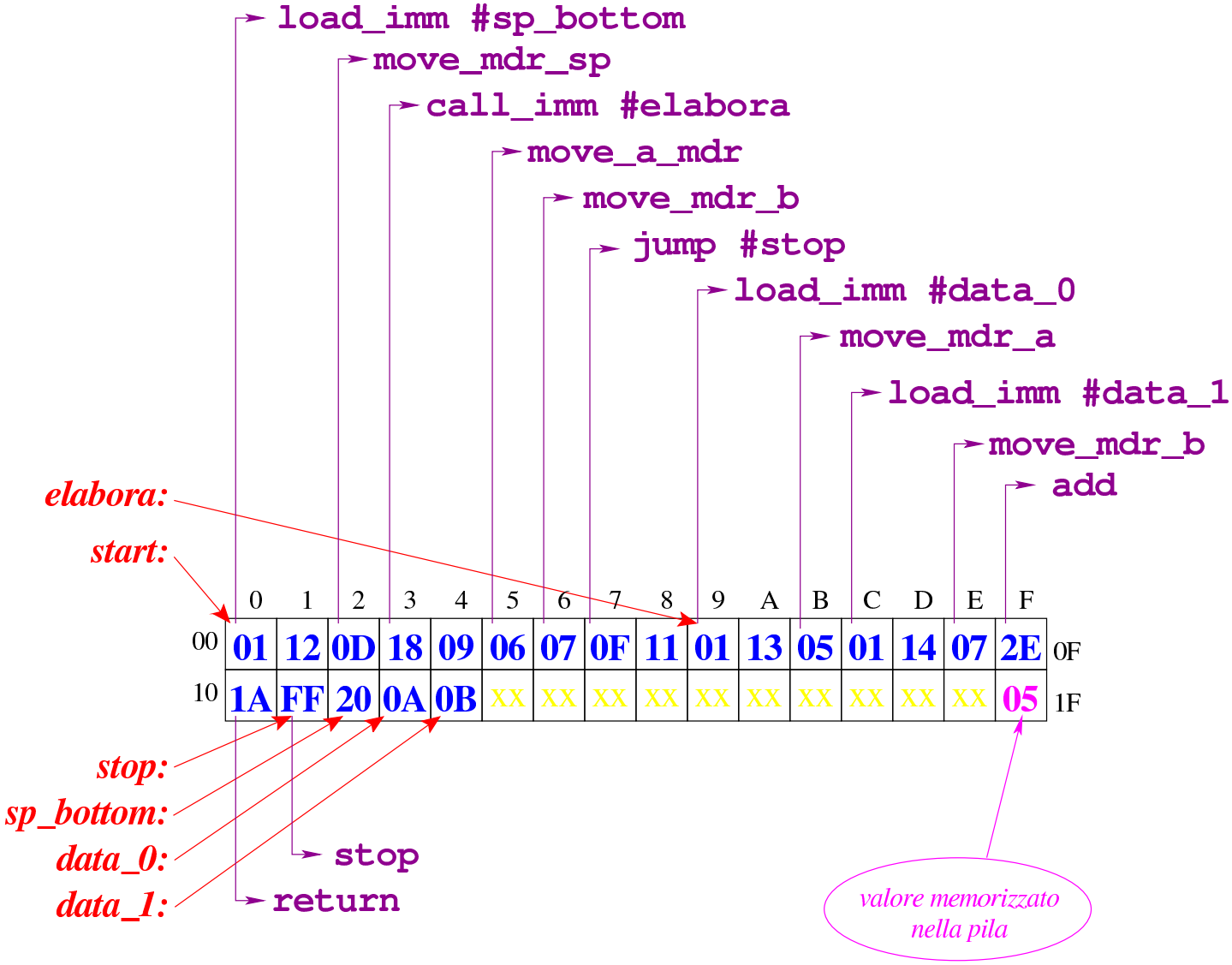


Figura u112.13. Situazione conclusiva del bus dati. Video: *http://www.youtube.com/watch?v=nWdXMvegkjc*