Nella quarta versione della CPU dimostrativa, viene aggiunta un'unità aritmetica, logica e di scorrimento (ALU), ma per il momento senza gestire gli indicatori (riporto, segno, zero e straripamento).

Figura u109.1. Il bus della CPU con l'aggiunta dell'unità ALU.

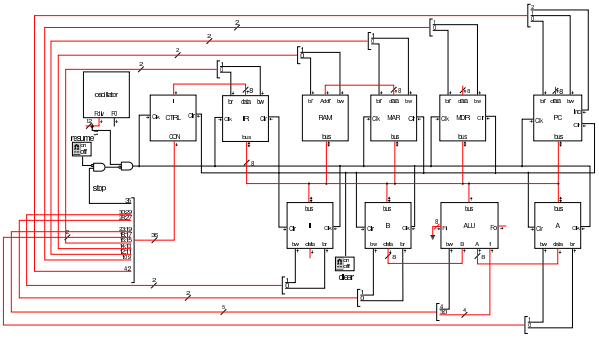
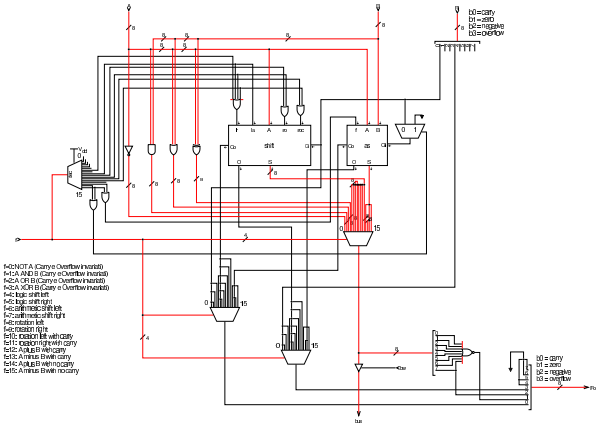


Figura u109.2. La struttura della ALU: si deve fare attenzione a non confondere le linee da un solo bit (di colore nero), rispetto a quelle che ne raccolgono in ranghi maggiori (di colore rosso).



«a2» 2013.11.11 --- Copyright © Daniele Giacomini -- appunti2@gmail.com http://informaticalibera.net

Figura u109.3. Modulo **shift** che si occupa di gestire gli scorrimenti e le rotazioni dei bit.

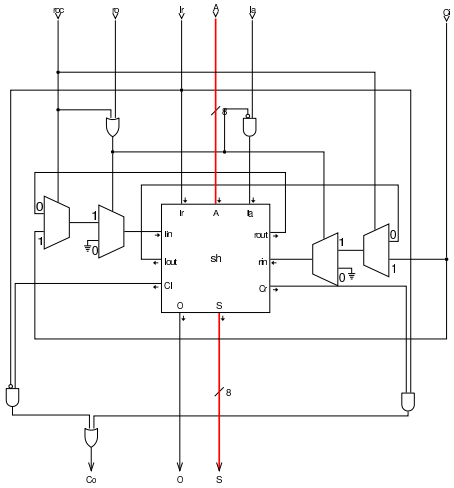


Figura u109.4. Modulo **sh**, contenuto nel modulo **shift**, per lo scorrimento dei bit.

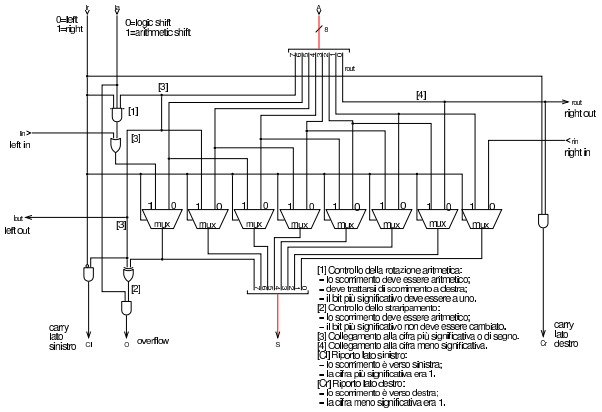
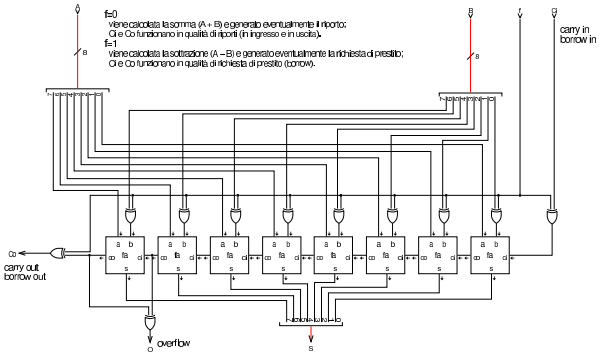


Figura u109.5. Modulo **as** della ALU che ha il compito di sommare o sottrarre gli ingressi.



Nel codice che descrive i campi del bus di controllo, si aggiungono quelli seguenti, i quali servono specificatamente a gestire la ALU. Si può osservare che la ALU ha il controllo di scrittura nel bus, ma non quello di lettura, dato che dal bus non riceve dati, e richiede il controllo della funzione che vi si vuole svolgere :

```
field alu_f[22:19]={
        not_a=0,
        a_and_b=1,
        a_or_b=2,
        a_xor_b=3,
        logic_shift_left=4,
        logic_shift_right=5,
        arith_shift_left=6,
        arith_shift_right=7,
        rotate_left=8,
        rotate_right=9,
        rotate_carry_left=10,
        rotate_carry_right=11,
        a_plus_b_carry=12,
        a_minus_b_borrow=13,
        a_plus_b=14,
        a_minus_b=15
};
field alu_bw[23];       // ALU --> bus
field fl_ar[24];        // FL <-- ALU
```


Tra i campi del bus di controllo si vede anche ***fl_ar*** che per ora può essere ignorato: viene chiarito il suo utilizzo quando nella prossima versione della CPU dimostrativa si aggiunge il registro ***FL***. Attualmente, nel microcodice vi si fa già riferimento, perché le microistruzioni prese ora in considerazione, in un secondo momento devono avere a che fare con tale registro.

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

```
op not {
  map not : 32;                         // A = NOT A
  +0[7:0]=32;
  operands op_0;
};
op and {
  map and : 33;                         // A = A AND B
  +0[7:0]=33;
  operands op_0;
};
op or {
  map or : 34;                          // A = A OR B
  +0[7:0]=34;
  operands op_0;
};
op xor {
  map xor : 35;                         // A = A OR B
  +0[7:0]=35;
  operands op_0;
};
op lshl {
  map lshl : 36;                        // A = A << 1
  +0[7:0]=36;
  operands op_0;
};
op lshr {
  map lshr : 37;                        // A = A >> 1
  +0[7:0]=37;
  operands op_0;
};
op ashl {
  map ashl : 38;                        // A = A << 1
  +0[7:0]=38;
  operands op_0;
};
op ashr {
  map ashr : 39;                        // A = +/-A >> 1
  +0[7:0]=39;
  operands op_0;
};
op rotl {
  map rotl : 40;                        // A = A rotate left
  +0[7:0]=40;
  operands op_0;
};
op rotr {
  map rotr : 41;                        // A = A rotate right
  +0[7:0]=41;
  operands op_0;
};
op add {
  map add : 46;                         // A = A + B
  +0[7:0]=46;
  operands op_0;
};
op sub {
  map sub : 47;                         // A = A - B
  +0[7:0]=47;
```

```
  operands op_0;
};
```

```
begin microcode @ 0
…
not:
    a_br alu_f=not_a alu_bw fl_ar;        // A <-- NOT A
    ctrl_start ctrl_load;                 // CNT <-- 0
//
and:
    a_br alu_f=a_and_b alu_bw fl_ar;      // A <-- A AND B
    ctrl_start ctrl_load;                 // CNT <-- 0
//
or:
    a_br alu_f=a_or_b alu_bw fl_ar;       // A <-- A OR B
    ctrl_start ctrl_load;                 // CNT <-- 0
//
xor:
    a_br alu_f=a_xor_b alu_bw fl_ar;      // A <-- A XOR B
    ctrl_start ctrl_load;                 // CNT <-- 0
//
lshl:
    a_br alu_f=logic_shift_left alu_bw fl_ar; // A <-- A << 1
    ctrl_start ctrl_load;                    // CNT <-- 0
//
lshr:
    a_br alu_f=logic_shift_right alu_bw fl_ar; // A <-- A >> 1
    ctrl_start ctrl_load;                     // CNT <-- 0
//
ashl:
    a_br alu_f=arith_shift_left alu_bw fl_ar; // A <-- A*2
    ctrl_start ctrl_load;                    // CNT <-- 0
//
ashr:
    a_br alu_f=arith_shift_right alu_bw fl_ar; // A <-- A/2
    ctrl_start ctrl_load;                     // CNT <-- 0
//
rotl:
    a_br alu_f=rotate_left alu_bw fl_ar; // A <-- A rot. left
    ctrl_start ctrl_load;                // CNT <-- 0
//
rotr:
    a_br alu_f=rotate_right alu_bw fl_ar; // A <-- A rot. right
    ctrl_start ctrl_load;                 // CNT <-- 0
//
add:
    a_br alu_f=a_plus_b alu_bw fl_ar;     // A <-- A + B
    ctrl_start ctrl_load;                 // CNT <-- 0
//
sub:
    a_br alu_f=a_minus_b alu_bw fl_ar;    // A <-- A - B
    ctrl_start ctrl_load;                 // CNT <-- 0
…
end
```

Figura u109.9. Corrispondenza con il contenuto della memoria che rappresenta il microcodice (la coppia ***m1*** e ***m2*** dell'unità di controllo).

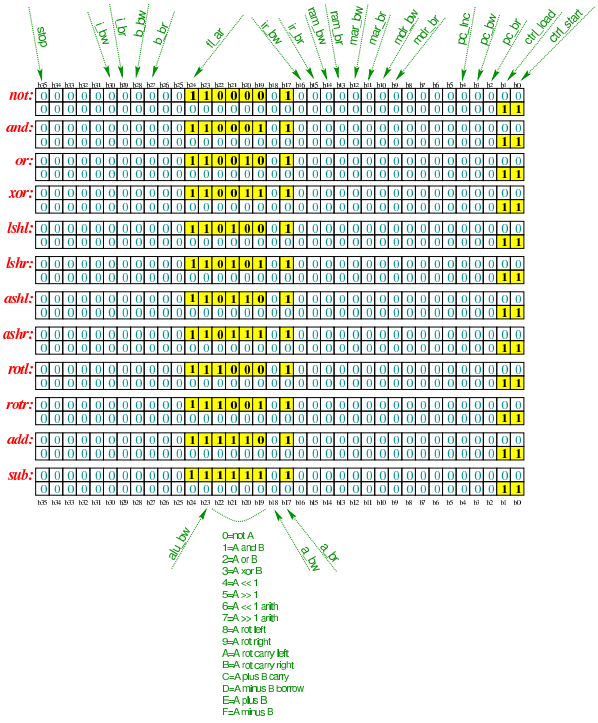


Tabella u109.10. Elenco delle macroistruzioni aggiunte in questa versione della CPU dimostrativa. Nella descrizione sintetica delle operazioni si usa la notazione del linguaggio C.

Sintassi	Descrizione
`not`	***A*** = ~***A*** Complemento a uno del contenuto di ***A***.
`and`	***A*** = ***A*** & ***B*** Si assegna ad ***A*** il risultato di ***A*** AND ***B***, bit per bit.
`or`	***A*** = ***A*** | ***B*** Si assegna ad ***A*** il risultato di ***A*** OR ***B***, bit per bit.
`xor`	***A*** = ***A*** ^ ***B*** Si assegna ad ***A*** il risultato di ***A*** XOR ***B***, bit per bit.
`lshl`	***A*** = ***A*** << 1 Si assegna ad ***A*** il risultato dello scorrimento logico a sinistra dei bit di ***A***.
`lshr`	***A*** = ***A*** >> 1 Si assegna ad ***A*** il risultato dello scorrimento logico a destra dei bit di ***A***.
`ashl`	***A*** = ***A*** << 1 Si assegna ad ***A*** il risultato dello scorrimento aritmetico a sinistra dei bit di ***A*** (in pratica è identico a **`lshl`**).
`ashr`	***A*** = ***A*** >> 1 Si assegna ad ***A*** il risultato dello scorrimento aritmetico a destra dei bit di ***A***.
`rotl`	Si assegna ad ***A*** il risultato della rotazione a sinistra dei bit di ***A***.
`rotr`	Si assegna ad ***A*** il risultato della rotazione a destra dei bit di ***A***.
`add`	***A*** = ***A*** + ***B*** Si assegna ad ***A*** il risultato della somma di ***A*** e ***B***, senza tenere conto del riporto precedente.

Sintassi	Descrizione
`sub`	***A*** = ***A*** − ***B*** Si assegna ad ***A*** il risultato della sottrazione di ***A*** e ***B***, senza tenere conto della richiesta di prestito precedente.

Nelle sezioni successive, vengono proposti diversi esempi, nei quali si sperimentano tutte le istruzioni nuove introdotte.

Istruzione «not»

Listato u109.11. Macrocodice per sperimentare l'istruzione **`not`**: si carica un valore dalla memoria, lo si copia nel registro ***A***, si calcola il complemento a uno e il risultato va ad aggiornare il registro ***A***. Il file completo che descrive le memorie per Tkgate dovrebbe essere disponibile presso *allegati/circuiti-logici/scpu-sub-d-not.gm* .

```
begin macrocode @ 0
start:
        load_imm #data_1
        move_mdr_a
        not
stop:
        stop
data_1:
        .byte 17
end
```

Figura u109.12. Contenuto della memoria RAM. Le celle indicate con «xx» hanno un valore indifferente.

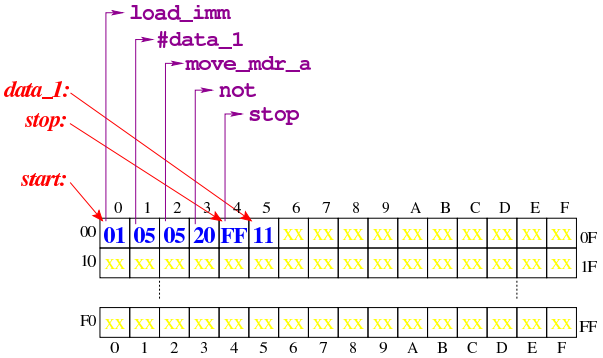
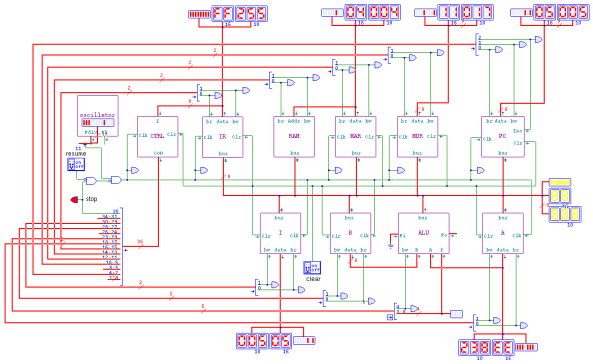


Figura u109.13. Situazione conclusiva del bus dati, dopo l'esecuzione dell'istruzione **`not`**. Video: *http://www.youtube.com/watch?v=x5Vnhd72vh728*



Istruzione «and»

Listato u109.14. Macrocodice per sperimentare l'istruzione **and**: si caricano dalla memoria i valori da assegnare ai registri ***A*** e ***B***, quindi si esegue un AND binario che va ad aggiornare il registro ***A***. Il file completo che descrive le memorie per Tkgate dovrebbe essere disponibile presso *allegati/circuiti-logici/scpu-sub-d-and.gm* .

```
begin macrocode @ 0
start:
        load_imm #data_1
        move_mdr_a
        load_imm #data_2
        move_mdr_b
        and
stop:
        stop
data_1:
        .byte 17
data_2:
        .byte 11
end
```

Figura u109.15. Contenuto della memoria RAM. Le celle indicate con «xx» hanno un valore indifferente.

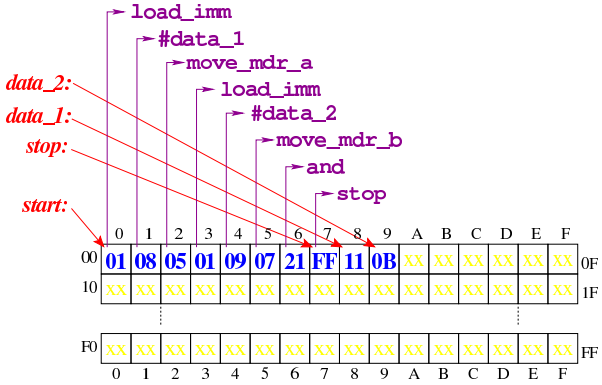
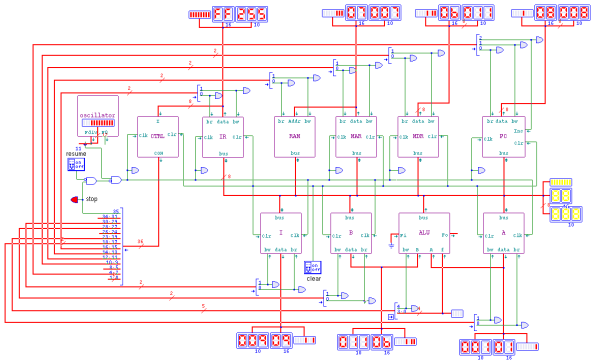


Figura u109.16. Situazione conclusiva del bus dati, dopo l'esecuzione dell'istruzione **and**. Video: *http://www.youtube.com/watch?v=2ra7SHxBvYY*



Istruzione «or»

Listato u109.17. Macrocodice per sperimentare l'istruzione **`or`**: si caricano dalla memoria i valori da assegnare ai registri ***A*** e ***B***, quindi si esegue un OR binario che va ad aggiornare il registro ***A***. Il file completo che descrive le memorie per Tkgate dovrebbe essere disponibile presso *allegati/circuiti-logici/scpu-sub-d-or.gm* .

```
begin macrocode @ 0
start:
        load_imm #data_1
        move_mdr_a
        load_imm #data_2
        move_mdr_b
        or
stop:
        stop
data_1:
        .byte 17
data_2:
        .byte 11
end
```

Figura u109.18. Contenuto della memoria RAM. Le celle indicate con «xx» hanno un valore indifferente.

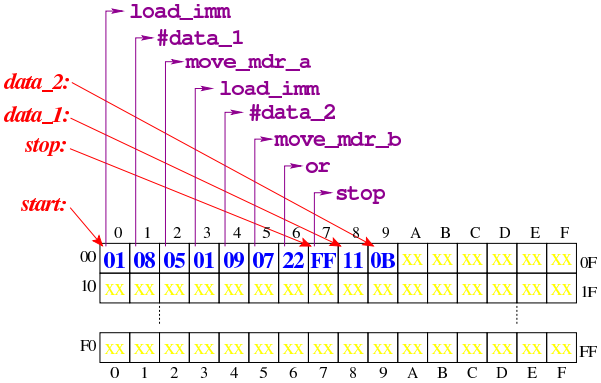
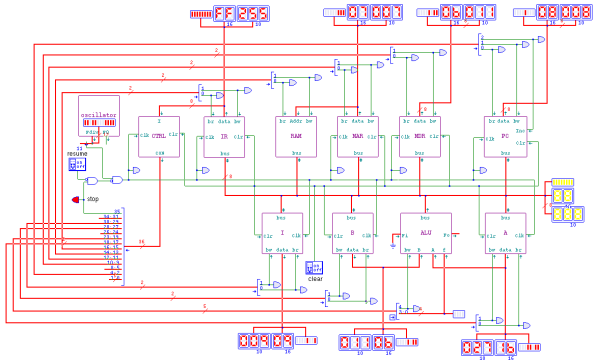


Figura u109.19. Situazione conclusiva del bus dati, dopo l'esecuzione dell'istruzione **or**. Video: *http://www.youtube.com/watch?v=7E-2uA6fVoY*



Istruzione «xor»

Listato u109.20. Macrocodice per sperimentare l'istruzione **xor**: si caricano dalla memoria i valori da assegnare ai registri ***A*** e ***B***, quindi si esegue un XOR binario che va ad aggiornare il registro ***A***. Il file completo che descrive le memorie per Tkgate dovrebbe essere disponibile presso *allegati/circuiti-logici/scpu-sub-d-xor.gm* .

```
begin macrocode @ 0
start:
        load_imm #data_1
        move_mdr_a
        load_imm #data_2
        move_mdr_b
        xor
stop:
        stop
data_1:
        .byte 17
data_2:
        .byte 11
end
```


Figura u109.21. Contenuto della memoria RAM. Le celle indicate con «xx» hanno un valore indifferente.

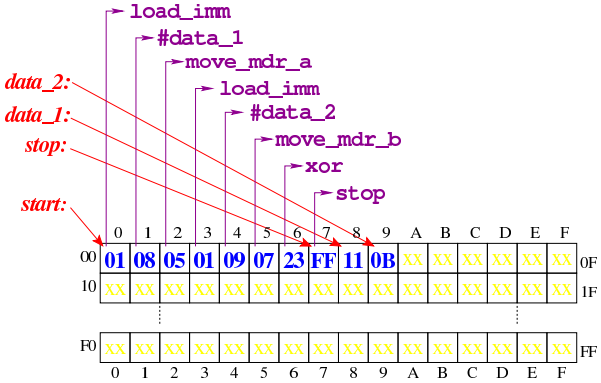
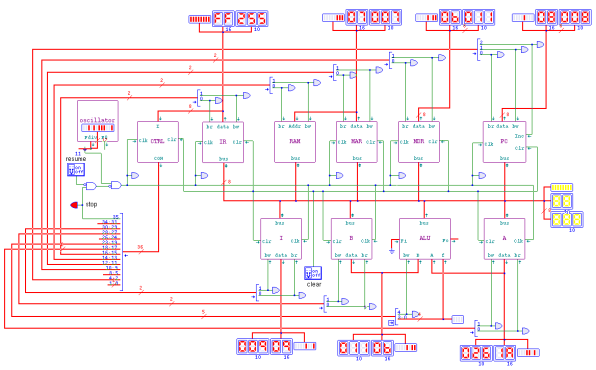


Figura u109.22. Situazione conclusiva del bus dati, dopo l'esecuzione dell'istruzione **xor**. Video: *http://www.youtube.com/watch?v=TuzknbyeabQ*



Istruzioni «lshl» e «lshr»

Listato u109.23. Macrocodice per sperimentare le istruzioni di scorrimento logico: si carica un valore dalla memoria, lo si copia nel registro ***A***, si esegue lo scorrimento a sinistra e il risultato va ad aggiornare il registro ***A***; si copia il risultato nel registro ***B*** e si carica nuovamente il valore originale per eseguire lo scorrimento a destra (che va ad aggiornare sempre il registro ***A***). Il file completo che descrive le memorie per Tkgate dovrebbe essere disponibile presso *allegati/circuiti-logici/scpu-sub-d-lsh.gm* .

```
begin macrocode @ 0
start:
        load_imm #data_1
        move_mdr_a
        lshl
        move_a_mdr
        move_mdr_b
        load_imm #data_1
        move_mdr_a
        lshr
stop:
        stop
data_1:
        .byte 17
end
```

Figura u109.24. Contenuto della memoria RAM. Le celle indicate con «xx» hanno un valore indifferente.

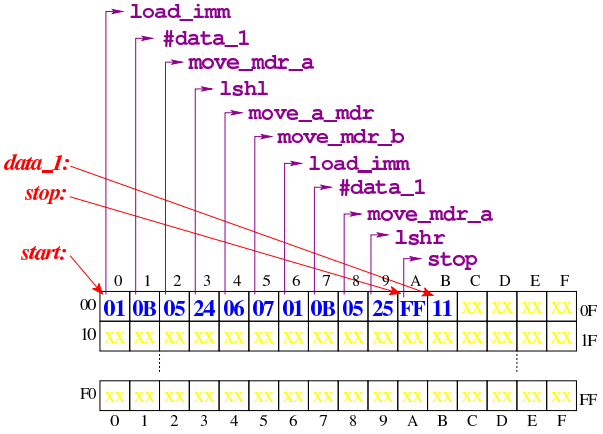
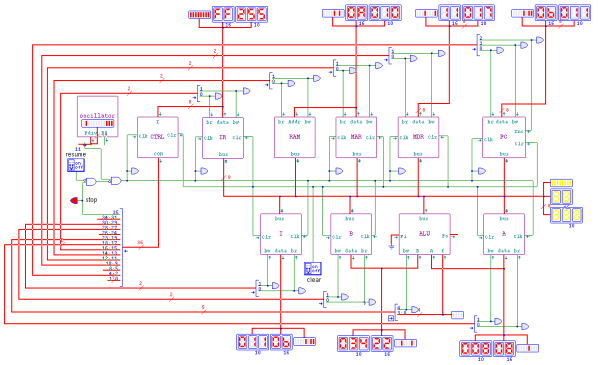


Figura u109.25. Situazione conclusiva del bus dati. Video: *http://www.youtube.com/watch?v=pkRfWYqGeB4*



Istruzioni «ashl» e «ashr»

Listato u109.26. Macrocodice per sperimentare le istruzioni di scorrimento aritmetico: si carica un valore dalla memoria, lo si copia nel registro ***A***, si esegue lo scorrimento a sinistra e il risultato va ad aggiornare il registro ***A***; si copia il risultato nel registro ***B*** e si carica nuovamente il valore originale per eseguire lo scorrimento a destra (che va ad aggiornare sempre il registro ***A***). Il file completo che descrive le memorie per Tkgate dovrebbe essere disponibile presso *allegati/circuiti-logici/scpu-sub-d-ash.gm* .

```
begin macrocode @ 0
start:
        load_imm #data_1
        move_mdr_a
        ashl
        move_a_mdr
        move_mdr_b
        load_imm #data_1
        move_mdr_a
        ashr
stop:
        stop
data_1:
        .byte 143
end
```


Figura u109.27. Contenuto della memoria RAM. Le celle indicate con «xx» hanno un valore indifferente.

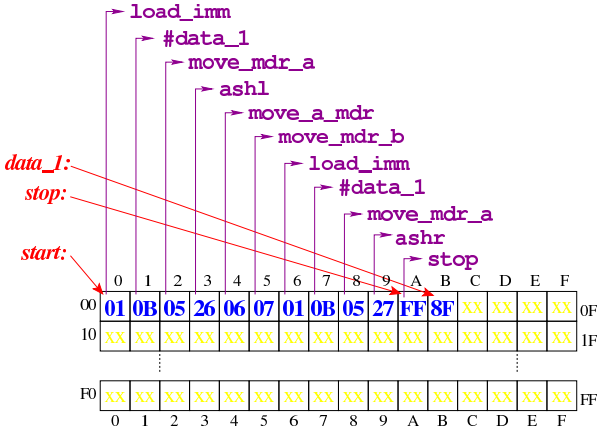
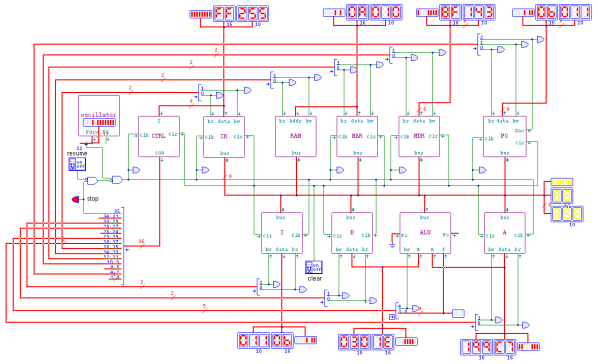


Figura u109.28. Situazione conclusiva del bus dati. Video: *http://www.youtube.com/watch?v=3rvR1WvWd1k*



Istruzioni «rotl» e «rotr»

Listato u109.29. Macrocodice per sperimentare le istruzioni di scorrimento logico: si carica un valore dalla memoria, lo si copia nel registro ***A***, si esegue la rotazione a sinistra e il risultato va ad aggiornare il registro ***A***; si copia il risultato nel registro ***B*** e si carica nuovamente il valore originale per eseguire la rotazione a destra (che va ad aggiornare sempre il registro ***A***). Il file completo che descrive le memorie per Tkgate dovrebbe essere disponibile presso *allegati/circuiti-logici/scpu-sub-d-rot.gm* .

```
begin macrocode @ 0
start:
        load_imm #data_1
        move_mdr_a
        rotl
        move_a_mdr
        move_mdr_b
        load_imm #data_1
        move_mdr_a
        rotr
stop:
        stop
data_1:
        .byte 17
end
```

Figura u109.30. Contenuto della memoria RAM. Le celle indicate con «xx» hanno un valore indifferente.

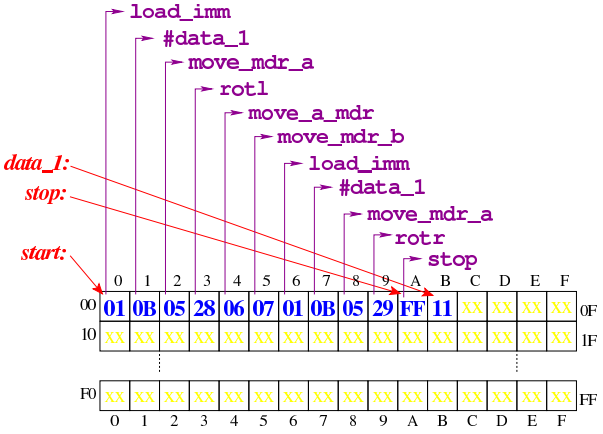
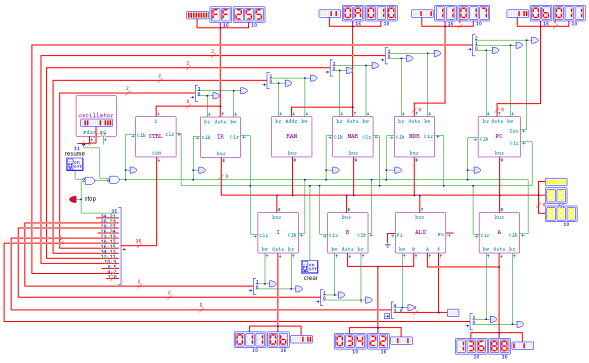


Figura u109.31. Situazione conclusiva del bus dati. Video: *http://www.youtube.com/watch?v=KCi8n6bnLQo*



Istruzione «add»

Listato u109.32. Macrocodice per sperimentare l'istruzione **add**: si caricano dalla memoria i valori da assegnare ai registri ***A*** e ***B***, quindi si esegue la somma che va ad aggiornare il registro ***A***. Il file completo che descrive le memorie per Tkgate dovrebbe essere disponibile presso *allegati/circuiti-logici/scpu-sub-d-add.gm* .

```
begin macrocode @ 0
start:
        load_imm #data_1
        move_mdr_a
        load_imm #data_2
        move_mdr_b
        add
stop:
        stop
data_1:
        .byte 17
data_2:
        .byte 11
end
```


Figura u109.33. Contenuto della memoria RAM. Le celle indicate con «xx» hanno un valore indifferente.

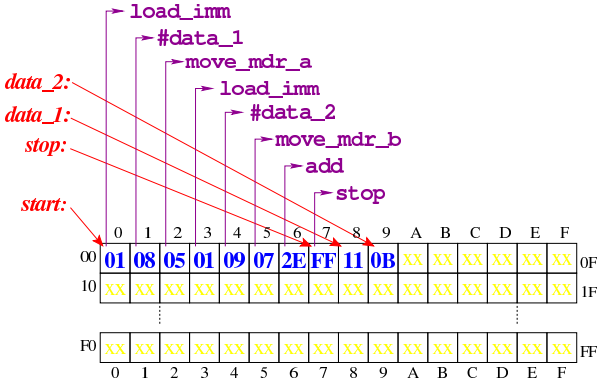
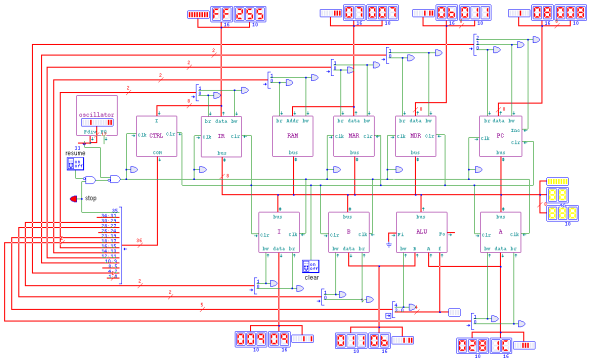


Figura u109.34. Situazione conclusiva del bus dati, dopo l'esecuzione dell'istruzione **add**. Video: *http://www.youtube.com/watch?v=QQJwz2yVwA8*



Istruzione «sub»

Listato u109.35. Macrocodice per sperimentare l'istruzione **sub**: si caricano dalla memoria i valori da assegnare ai registri ***A*** e ***B***, quindi si esegue la sottrazione (***A*−*B***) che va ad aggiornare il registro ***A***. Il file completo che descrive le memorie per Tkgate dovrebbe essere disponibile presso *allegati/circuiti-logici/scpu-sub-d-sub.gm* .

```
begin macrocode @ 0
start:
        load_imm #data_1
        move_mdr_a
        load_imm #data_2
        move_mdr_b
        sub
stop:
        stop
data_1:
        .byte 17
data_2:
        .byte 11
end
```

Figura u109.36. Contenuto della memoria RAM. Le celle indicate con «xx» hanno un valore indifferente.

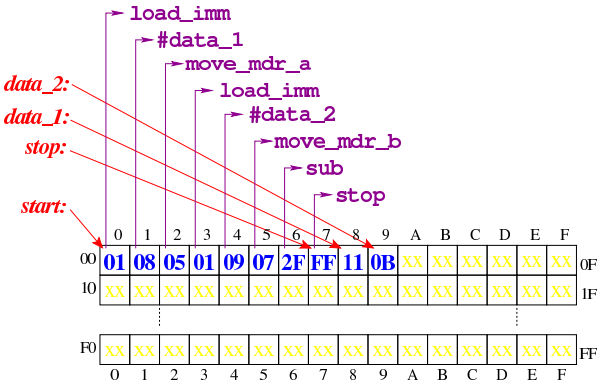


Figura u109.37. Situazione conclusiva del bus dati, dopo l'esecuzione dell'istruzione **sub**. Video: *http://www.youtube.com/watch?v=VRd8ilJbK_Y*