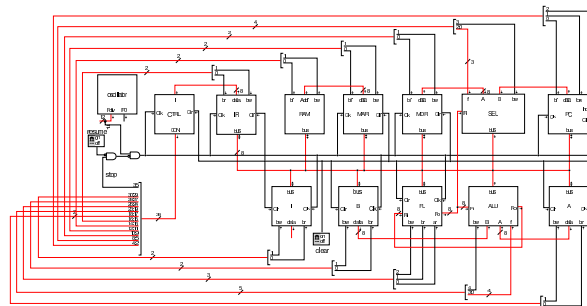


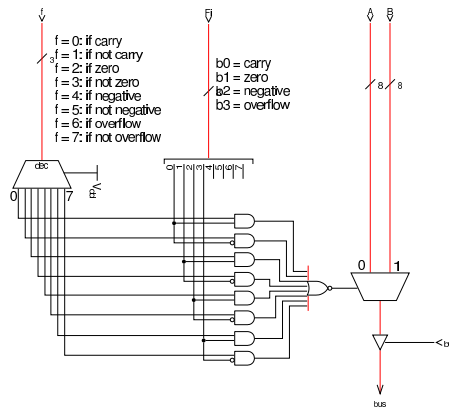
Nella sesta versione della CPU dimostrativa, viene aggiunto un modulo che consente di eseguire delle comparazioni, sulla base dello stato degli indicatori annotati nel registro *FL*, scegliendo tra due valori che in questo progetto sono costituiti dal contenuto del registro *PC* o dal contenuto del registro *MDR*.

Figura u111.1. Il bus della CPU con l'aggiunta del modulo *SEL* per la gestione condizioni.



Il modulo *SEL* riceve due valori dagli ingressi *A* e *B*; dall'ingresso *Fi* riceve lo stato degli indicatori, così come emesso dal registro *FL*. Sulla base della funzione che si seleziona attraverso l'ingresso *f*, quando è attivo l'ingresso *bw*, il modulo immette nel bus uno dei due valori disponibili negli ingressi *A* e *B*. Quando la condizione rappresentata dalla funzione si avvera, viene scelto il valore dell'ingresso *A*, altrimenti si prende *B*.

Figura u111.2. La struttura interna del modulo *SEL*.



«02-2013.11.11 ... Copyright © Daniele Giacomini - appunti2@gmail.com http://informaticalibera.net

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

```

field sel_f[7:5]={
    if_carry=0,
    if_not_carry=1,
    if_zero=2,
    if_not_zero=3,
    if_negative=4,
    if_not_negative=5,
    if_overflow=6,
    if_not_overflow=7
};
field sel_bw[8]; // SEL --> bus
    
```

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

```

op jump_if_carry {
    map jump_if_carry : 16; // jump to #nn if carry=1
    +0[7:0]=16;
    operands op_1;
}
    
```

```

};
op jump_if_not_carry {
  map jump_if_not_carry : 17; // jump to #nn if carry==0
  +0[7:0]=17;
  operands op_1;
};
op jump_if_zero {
  map jump_if_zero : 18; // jump to #nn if zero==1
  +0[7:0]=18;
  operands op_1;
};
op jump_if_not_zero {
  map jump_if_not_zero : 19; // jump to #nn if zero==0
  +0[7:0]=19;
  operands op_1;
};
op jump_if_negative {
  map jump_if_negative : 20; // jump to #nn if negative==1
  +0[7:0]=20;
  operands op_1;
};
op jump_if_not_negative {
  map jump_if_not_negative : 21; // jump to #nn if negative==0
  +0[7:0]=21;
  operands op_1;
};
op jump_if_overflow {
  map jump_if_overflow : 22; // jump to #nn if overflow==1
  +0[7:0]=22;
  operands op_1;
};
op jump_if_not_overflow {
  map jump_if_not_overflow : 23; // jump to #nn if overflow==0
  +0[7:0]=23;
  operands op_1;
};
};

```

```

begin microcode @ 0
...
jump_if_carry:
  mar_br pc_bw; // MAR <-- PC
  pc_inc; // PC++
  mdr_br ram_bw; // MDR <-- RAM[mar]
  pc_br sel_f=if_carry sel_bw; // PC = (carry ? MAR : PC)
  ctrl_start ctrl_load; // CNT <-- 0
//
jump_if_not_carry:
  mar_br pc_bw; // MAR <-- PC
  pc_inc; // PC++
  mdr_br ram_bw; // MDR <-- RAM[mar]
  pc_br sel_f=if_not_carry sel_bw; // PC = (not_carry ? MAR : PC)
  ctrl_start ctrl_load; // CNT <-- 0
//
jump_if_zero:
  mar_br pc_bw; // MAR <-- PC
  pc_inc; // PC++
  mdr_br ram_bw; // MDR <-- RAM[mar]
  pc_br sel_f=if_zero sel_bw; // PC = (zero ? MAR : PC)
  ctrl_start ctrl_load; // CNT <-- 0
//
jump_if_not_zero:
  mar_br pc_bw; // MAR <-- PC
  pc_inc; // PC++
  mdr_br ram_bw; // MDR <-- RAM[mar]
  pc_br sel_f=if_not_zero sel_bw; // PC = (not_zero ? MAR : PC)
  ctrl_start ctrl_load; // CNT <-- 0
//
jump_if_negative:
  mar_br pc_bw; // MAR <-- PC
  pc_inc; // PC++
  mdr_br ram_bw; // MDR <-- RAM[mar]
  pc_br sel_f=if_negative sel_bw; // PC = (negative ? MAR : PC)
  ctrl_start ctrl_load; // CNT <-- 0
//
jump_if_not_negative:
  mar_br pc_bw; // MAR <-- PC
  pc_inc; // PC++
  mdr_br ram_bw; // MDR <-- RAM[mar]
  pc_br sel_f=if_not_negative sel_bw; // PC = (not_negative ? MAR : PC)
  ctrl_start ctrl_load; // CNT <-- 0
//
jump_if_overflow:
  mar_br pc_bw; // MAR <-- PC
  pc_inc; // PC++
  mdr_br ram_bw; // MDR <-- RAM[mar]
  pc_br sel_f=if_overflow sel_bw; // PC = (overflow ? MAR : PC)
  ctrl_start ctrl_load; // CNT <-- 0
//
jump_if_not_overflow:
  mar_br pc_bw; // MAR <-- PC
  pc_inc; // PC++

```

```

mdr_br ram_bw; // MDR <-- RAM[mar]
pc_br sel_f=if_not_overflow sel_bw; // PC = (not_overflow ? MAR : PC)
ctrl_start ctrl_load; // CNT <-- 0
...
end

```

Figura u11.6. Corrispondenza con il contenuto della memoria che rappresenta il microcodice (la coppia *m1* e *m2* dell'unità di controllo).

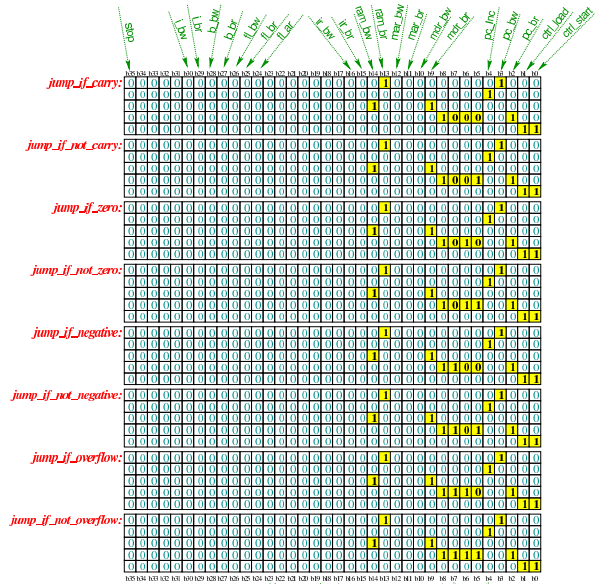


Tabella u11.7. Elenco delle macrostrutture aggiunte in questa versione della CPU dimostrativa.

Sintassi	Descrizione
jump_if_carry <i>indirizzo</i>	Salta all'indirizzo specificato se l'indicatore di riporto è attivo.
jump_if_not_carry <i>indirizzo</i>	Salta all'indirizzo specificato se l'indicatore di riporto non è attivo.
jump_if_zero <i>indirizzo</i>	Salta all'indirizzo specificato se l'indicatore zero è attivo.
jump_if_not_zero <i>indirizzo</i>	Salta all'indirizzo specificato se l'indicatore zero non è attivo.
jump_if_negative <i>indirizzo</i>	Salta all'indirizzo specificato se l'indicatore di valore negativo è attivo.
jump_if_not_negative <i>indirizzo</i>	Salta all'indirizzo specificato se l'indicatore di valore negativo non è attivo.
jump_if_overflow <i>indirizzo</i>	Salta all'indirizzo specificato se l'indicatore di straripamento è attivo.
jump_if_not_overflow <i>indirizzo</i>	Salta all'indirizzo specificato se l'indicatore di straripamento non è attivo.

Listato u11.8. Macrocodice per sperimentare l'uso del modulo di selezione, nel quale si crea un ciclo che incrementa una variabile, di una unità alla volta, fino a quando questa variabile contiene il risultato della somma con un'altra. Il file completo che descrive le memorie per Tkgate dovrebbe essere disponibile presso [allegati/circuiti-logici/scpu-sub-f-jump-if-gm](#).

```

begin macrocode @ 0
start:
  load_imm #costante_zero
  move_mdr_b

```

```

load_imm #variabile_x
move_mdr_a
add

ciclo:
jump_if_zero #stop
load_imm #variabile_y
move_mdr_a
load_imm #costante_uno
move_mdr_b
add
move_a_mdr
store_imm #variabile_y
load_imm #variabile_x
move_mdr_a
load_imm #costante_uno
move_mdr_b
sub
move_a_mdr
store_imm #variabile_x
jump #ciclo

stop:
stop

costante_zero:
.byte 0
costante_uno:
.byte 1
variabile_x:
.byte 3
variabile_y:
.byte 7
end

```

Figura u111.10. Situazione conclusiva del bus dati. Video: <http://www.youtube.com/watch?v=hFoOoGf86t0>

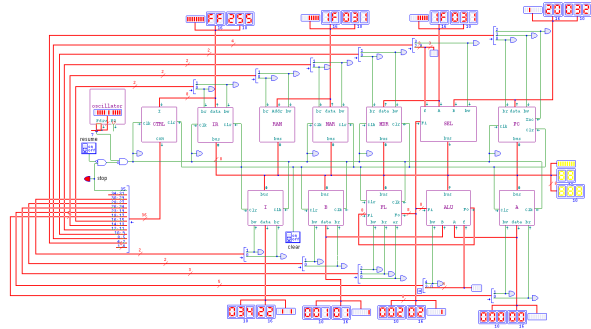


Figura u111.9. Contenuto della memoria RAM prima dell'esecuzione. Le celle indicate con «xx» hanno un valore indifferente. Al termine dell'esecuzione, la cella di memoria all'indirizzo 23_{16} , corrispondente a *variabile_y*, contiene il valore 10 ($0A_{16}$).

