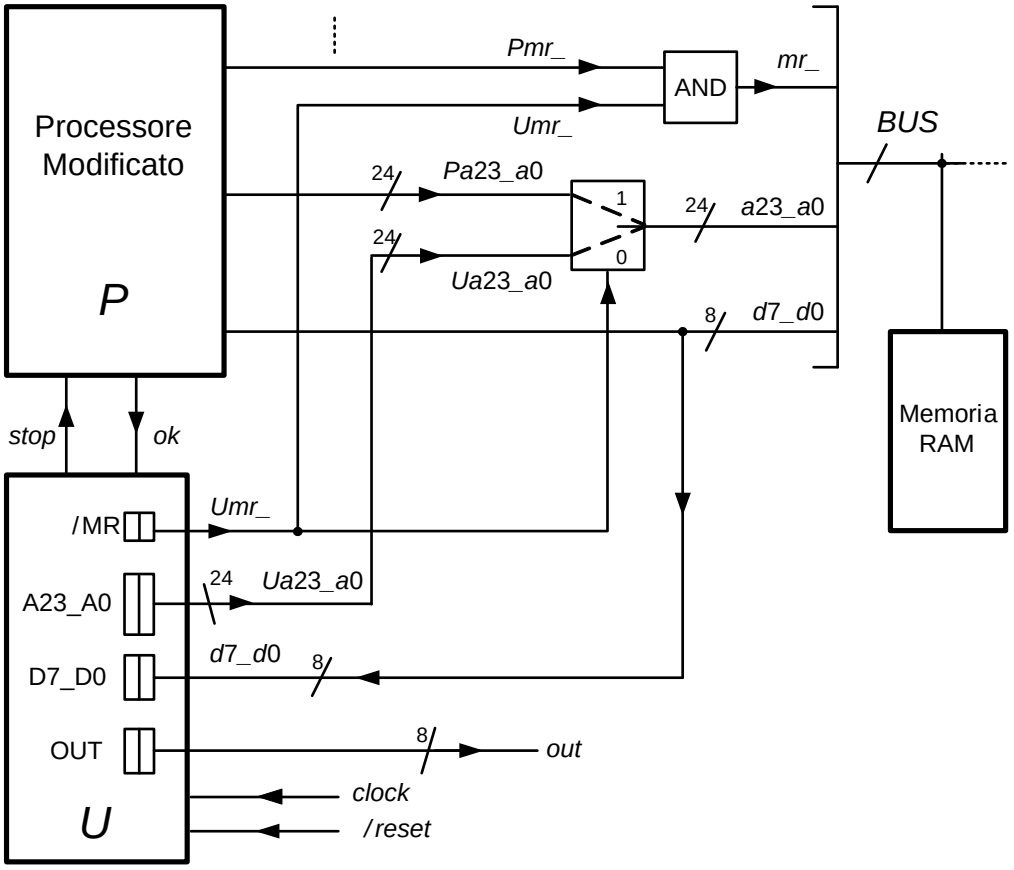
Esercizio 1

Dato un numero intero $a \in [-50; +49]$

1) Individuare il numero minimo di cifre n_{10} ed n_2 su cui a è rappresentabile in complemento alla radice in base 10 e in base 2.
2) Siano A_{10} ed A_2 le rappresentazioni in CR di a in base dieci e due, su n_{10} ed n_2 cifre, ed $\alpha_{n_{10}-1}, \dots, \alpha_0$ le cifre di A_{10}. Esprimere la relazione algebrica che lega A_2 alle cifre in base dieci, $A_2 = f(\alpha_{n_{10}-1}, \dots, \alpha_0)$.
3) Basandosi sul risultato del punto precedente, sintetizzare un circuito che ha in ingresso le cifre di A_{10} ed in uscita quelle di A_2.

Esercizio 2



Al Processore visto a lezione sono state aggiunte la variabile di ingresso *stop* e la variabile di uscita *ok* (inizializzata a 0 al reset), aventi le seguenti funzioni: quando *stop* viene messo a 1 dall'Unità *U*, il Processore, entro un tempo inferiore alla decina di cicli di clock ma non noto a priori, cessa di compiere qualunque azione e pone *ok* a 1. Quando l'Unità *U* riporta *stop* a 0, il Processore, entro un tempo inferiore alla decina di cicli di clock ma non noto a priori, pone *ok* a 0 e riprende la sua normale evoluzione.

Descrivere l'Unità *U* in modo all'infinito emetta, uno dopo l'altro, i byte che si trovano in memoria a partire dall'indirizzo 0. I byte debbono essere emessi tramite la variabile *out* e ogni byte deve permanere pe 100 cicli di clock, ma il Processore deve essere bloccato per il minimo tempo possibile. Si supponga la memoria sufficientemente veloce da non dover inserire stati di wait. Non si cambino i nomi delle variabili e dei registri indicati in figura. Disegnare il circuito della parte operativa relativo al registro OUT.

Esercizio 1 - Soluzione

Si ottiene $n_2 = \lceil log_2 50 \rceil = 7$ bit ed $n_{10} = \lceil log_{10} 50 \rceil = 2$ cifre. Le leggi di rappresentazione in CR sono:

$$L: \quad A = \begin{cases} a & a \geq 0 \\ \beta^n + a & a < 0 \end{cases} \qquad L^{-1}: \quad a = \begin{cases} A & a_{n-1} < \frac{\beta}{2} \\ A - \beta^n & a_{n-1} \geq \frac{\beta}{2} \end{cases}$$

Istanziando $\beta = 10$ ed $n = n_{10} = 2$ ed usando la legge inversa si ottiene:

$$a = \begin{cases} A_{10} & \alpha_1 < 5 \\ A_{10} - 100 & \alpha_1 \geq 5 \end{cases}$$

Dove α_1 è la cifra più significativa di A_{10}. Usando la legge diretta in base $\beta = 2$ ed $n = n_2 = 7$ si ottiene:

$$A_2 = \begin{cases} a & a \geq 0 \\ 128 + a & a < 0 \end{cases}$$

Mettendo le due relazioni insieme si ottiene quindi:

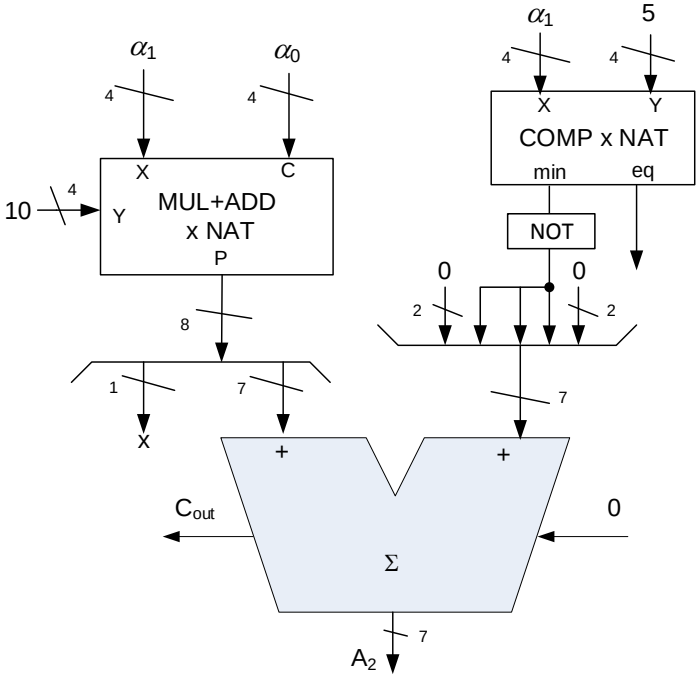
$$A_2 = \begin{cases} A_{10} & \alpha_1 < 5 \\ A_{10} + 28 & \alpha_1 \geq 5 \end{cases}$$

Dato che $A_{10} = 10 \cdot \alpha_1 + \alpha_0$, otteniamo:

$$A_2 = \begin{cases} \alpha_1 \cdot 10 + \alpha_0 & \alpha_1 < 5 \\ \alpha_1 \cdot 10 + \alpha_0 + 28 & \alpha_1 \geq 5 \end{cases}$$

La sintesi del circuito, riportata nella figura, implementa la relazione scritta sopra. L'uscita del comparatore genera le costanti 0 o 28 (= 16 + 8 + 4) su 7 bit. Sono possibili ottimizzazioni, tra cui:

- Sostituzione del moltiplicatore per 10 con sommatori (in base all'equivalenza $X \cdot 10 = X \cdot 8 + X \cdot 2$ ed al fatto che le moltiplicazioni per potenze di due sono shift sinistri);
- Sostituzione del comparatore con una rete combinatoria ottimizzata a due livelli di logica;
- Riduzione del numero di stadi del sommatore da 7 a 5 (si può osservare facilmente che le due cifre meno significative di A_2 sono sempre quelle in uscita dal moltiplicatore).



Esercizio 2 – soluzione

```
module U(d7_d0,Ua23_a0,Umr_,stop,ok,out,clock,reset_);
 input          clock,reset_;
 input[7:0]   d7_d0;
 output[23:0] Ua23_a0;
 output       Umr_;
 output       stop;
 input        ok;
 output[7:0]  out;
 reg           MR_,STOP;    assign Umr_=MR_; assign stop=STOP;
 reg[23:0]     A23_A0;      assign Ua23_a0=A23_A0;
 reg[7:0]      D7_D0,OUT;   assign out=OUT;
 reg[7:0]      COUNT; parameter num_clock=100;
 reg[2:0]      STAR;  parameter S0=0,S1=1,S2=2,S3=3,S4=4;
 always @(reset_==0)#1 begin MR_=1; A23_A0<=0; STOP=0; COUNT<=num_clock; STAR=S0;
end
 always @(posedge clock) if (reset_==1)#3
  casex(STAR)
   S0: begin COUNT<=COUNT-1; STOP<=1; STAR<=(ok==1)?S1:S0; end
   S1: begin COUNT<=COUNT-1; MR_<=0; STAR<=S2; end
   S2: begin COUNT<=COUNT-1; D7_D0<=d7_d0; A23_A0<=A23_A0+1; STOP<=0; MR_<=1;
             STAR<=S3; end
   S3: begin COUNT<=COUNT-1; STAR<=(ok==0)?S4:S3; end
   S4: begin COUNT<=(COUNT==1)?num_clock:COUNT-1; OUT<=(COUNT==1)?D7_D0:OUT;
             STAR<=(COUNT==1)?S0:S4; end
  endcase
endmodule
```